Refocusing generalised normalisation

José Espírito Santo*

Departamento de Matemática
Universidade do Minho
Portugal
jes@math.uminho.pt

Abstract. When defined with general elimination/application rules, natural deduction and λ-calculus become closer to sequent calculus. In order to get real isomorphism, normalisation has to be defined in a "multiary" variant, in which reduction rules are necessarily non-local (reason: normalisation, like cut-elimination, acts at the *head* of applicative terms, but natural deduction focuses at the *tail* of such terms). Non-local rules are bad, for instance, for the mechanization of the system. A solution is to extend natural deduction even further to a *unified calculus* based on the unification of cut and general elimination. In the unified calculus, a sequent term behaves like in the sequent calculus, whereas the reduction steps of a natural deduction term are interleaved with explicit steps for bringing heads to focus. A variant of the calculus has the symmetric role of improving sequent calculus in dealing with tail-active permutative conversions.
Keywords: normalisation, generalised elimination rules, multiarity

1 Introduction

Natural deduction with general elimination rules is closer to sequent calculus than traditional natural deduction [10]. This paper investigates the exact realization of this claim, and the outcome of such realization, in the context of intuitionistic implicational logic.

In [10] von Plato obtains a perfect correspondence between "fully normal" deductions and cut-free sequent derivation, extending to a *bijection* between natural deductions and a subset of sequent derivation (where cuts are necessarily "right-principal"). In this paper we start by investigating the *isomorphism* between cut-elimination and generalised normalisation. The systems are presented as typed λ-calculi, and a computational reading is present throughout.

Sequent calculus is presented as system λGm, where cuts are "right-principal". In λGm there is the so-called *multiarity* facility, *i.e.* the facility of not naming an active, linearly left-introduced formula [9]. In this system cuts correspond to applicative terms, consisting of a head, a list of arguments, and a "continuation" (or tail).

* The author is supported by FCT through the Centro de Matemática da Universidade do Minho.

In order to get real isomorphism, natural deduction is defined as the system λNm, a multiary extension of von Plato's system. However, normalisation has to be defined with non-local reduction rules. The reason is simple: normalisation, like cut-elimination, acts at the *head* of applicative terms, but natural deduction focuses at the *tail* of such terms. Now, in a multiary system, heads are arbitrarily distant from tails. So, we get isomorphism, but normalisation in the relevant natural deduction system is just a clumsy way of doing cut-elimination. Symmetrically, sequent calculus is the wrong setting for doing tail-active permutative conversions.

A way out of this situation, which is simultaneously the main outcome of proving $\lambda Gm \cong \lambda Nm$, suggested by the analysis of this isomorphism, is to extend natural deduction even further, to a calculus λU that *unifies* λGm and λNm. This *unified calculus* is based on the unification of cut and general elimination.

In the unified calculus all reduction rules are local, and a sequent term behaves like in the sequent calculus, whereas the reduction steps of a natural deduction term are interleaved with explicit steps for bringing heads to focus. This gives an implementation of multiary normalisation with local reduction steps.

The unified calculus seems particularly appropriate for dealing with conversions with are both head-acting and tail-acting. A variant of the calculus is suggested which has the role of improving sequent calculus in dealing with tail-acting permutative conversions.

Structure of the paper: Section 2 presents λGm, λNm and $\lambda Gm \cong \lambda Nm$. Section 3 is the central contribution, presenting the unified calculus λU and its properties and variants. Section 4 concludes.

Notations: Types (=formulas) are ranged over by A, B, C and generated from type variables using the "arrow type" (=implication), written $A \supset B$. Contexts Γ are consistent sets of declarations $x : A$. "Consistent" means that for each variable x there is at most one declaration in Γ. The notation $\Gamma, x : A$ always denotes a consistent union, that is, one that produces a consistent set. Barendregt's variable convention is adopted. In particular, we take renaming of bound variables for granted. Substitution is denoted by $[\_/x]\_$. "s.n." abbreviates "strongly normalising".

2 Isomorphism

Natural deduction with general elimination rules: This system [10] may be presented as a type system for the λ-calculus with generalised application. The latter is the system ΛJ of [6], which we rename here as λg, for the sake of uniformity with the names of other calculi. Terms of λg are given by

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

The typing rule for generalised application is

$$\frac{\Gamma \vdash M : A \supset B \quad \Gamma \vdash N : A \quad \Gamma, x : B \vdash P : C}{\Gamma \vdash M(N, x.P) : C}\ gElim$$

The λg-calculus has two reduction rules

$$\begin{array}{ll}(\beta) & (\lambda x.M)(N, y.P) \to [[N/x]M/y]P \\ (\pi) & M(N, x.P)(N', y.P') \to M(N, x.P(N', y.P'))\ .\end{array}$$

The usual λ-calculus embeds in λg by setting $MN = M(N, x.x)$. Likewise, *modus ponens* (=Gentzen's elimination rule for implication) may be seen as the particular case of $gElim$ where $B = C$ and the rightmost premiss is omitted.

Sequent calculus: We present the system λGm ("G" is after Gentzen). It should be understood as an extension of the λ-calculus whose typing rules define a sequent calculus. As an extension of the λ-calculus, it adds to the application constructor the features of generality and multiarity, to be explained soon. As a sequent calculus, λGm contains, as primitive, cuts of a special form, namely those whose cut-formula in the right premiss is principal in a left-introduction.

There are two sorts of expressions in λGm:

$$\begin{array}{ll}\text{(Terms)} & t, u, v ::= x \mid \lambda x.t \mid tl \\ \text{(Lists)} & l ::= u \cdot (x)v \mid u :: l\end{array}$$

A term of the form tl is called a *cut*. In general, l has the form $u_1 :: ... :: u_{n-1} :: u_n \cdot (x)v$, for some $n \geq 1$. We regard these expressions as *generalised lists*. In tl, think of t as a function and of l as an expression that provides a non-empty list of arguments for t (this is the multiarity feature), plus some "continuation" $(x)v$, specifying what to do after the last argument is consumed (this is the generality feature).

Define $[u] = u \cdot (x)x$. Expressions of the form $u_1 :: ... :: u_{n-1} :: [u_n]$ are regarded as lists, writen as $[u_1, ..., u_{n-1}, u_n]$, and interpreted in [5] as "applicative contexts". Lists in λGm may be interpreted as generalised applicative contexts.

There are two sorts of sequents in λGm, namely $\Gamma \vdash t : A$ and $\Gamma; A \vdash l : B$. The distinguished position in the antecedent of sequents of the latter kind is named the *stoup*. Typing rules are as follows:

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

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \supset B}\ Right \qquad \frac{\Gamma \vdash u : A \quad \Gamma; B \vdash l : C}{\Gamma; A \supset B \vdash u :: l : C}\ plLeft$$

$$\frac{\Gamma \vdash t : A \quad \Gamma; A \vdash l : B}{\Gamma \vdash tl : B}\ Cut \qquad \frac{\Gamma \vdash u : A \quad \Gamma, x : B \vdash v : C}{\Gamma; A \supset B \vdash u \cdot (x)v : C}\ gLeft$$

These rules define a sequent calculus, with a primitive rule of cut, and two sorts of primitive left-introduction rules: general left-introduction ($gLeft$) and principal-linear left-introduction ($plLeft$). The system is such that, in any derivation, the stoup always contains a formula that is *principal and linear*, that is, principal in a left-introduction rule, and introduced without contraction. Given that the cut rule and the left-introduction rules require some of the active formulas and/or

the principal formula to be in the stoup, these inference rules are of a particular kind. The cut-formula is always an implication and cut is right-principal. As to left-introduction rules, they are both linear, in the sense that they both introduce without contraction. In addition, rule $plLeft$, by requiring its right active formula to be principal and linear, is of the restricted form identified in [5, 1, 7].

There are three reduction rules

$$\begin{array}{lll} (\beta 1)\ (\lambda x.t)(u \cdot (y)v) \to [[u/x]t/y]v & & (\pi)\ (tl)l' \to t(l@l') \\ (\beta 2) \quad (\lambda x.t)(u :: l) \to ([u/x]t)l & & \end{array}$$

where $l@l'$ is the "append" of generalised lists l and l', defined by

$$(u :: l)@l' = u :: (l@l') \qquad (u \cdot (x)v)@l' = u \cdot (x)(vl') \ .$$

Let $\beta = \beta 1 \cup \beta 2$. By *cut-elimination* we mean $\beta\pi$-reduction. A $\beta\pi$-nf is a term where every cut has the form xl. These normal forms correspond exactly to the multiary sequent terms of [9]. In λGm, a λg-term is a term without occurrences of $u :: l$ (hence every cut in a λg-term is a g-application $t(u \cdot (x)v)$).

Multiary natural deduction: We present the system λNm. It should be understood as an extension of the λg-calculus and of natural deduction with general elimination rules. This system has an implementation of the multiarity feature (the ability of forming chains of arguments for a function) within the framework of natural deduction.

Expressions in λNm are given by:

$$\begin{array}{ll} \text{(Terms)} & M, N, P ::= x \mid \lambda x.M \mid app(F, N, (x)P) \\ \text{(Functions)} & F ::= hd(M) \mid FN \end{array}$$

This is a syntax with two syntactic classes: terms and *functions*. A term of the form $app(F, N, (x)P)$ may be called either *gm-application* or *outer application*. Think of this construction as an extension of the generalized application of λg. Indeed, generalized application is recovered as $app(hd(M), N, (x)P)$, because any term M can be coerced to a function $hd(M)$. There is a second kind of application construction, FN, named *inner application* or *mp-application*. Here *mp* is mnemonic of *modus ponens*.

The elements of the second syntactic class are named functions because, in expressions, they only occur in the function position of applications. Application FN is inner because, being a function, occurs in the function position of another application. The general form of a function is $hd(M)N_1...N_{m-1}$, for some $m \geq 1$. A function of the form $hd(M)$ is called a *head*. An intuition about functions is that they are expressions which require an immediate and linear use. On the other hand, in $app(F, N, (x)P)$, the use of the application of F to N, specified by the "continuation" $(x)P$ is required neither to be immediate nor linear.

There are two sorts of sequents in λNm, namely $\Gamma \vdash M : A$ and $\Gamma \rhd F : A$. Typing rules are as follows:

$$\frac{}{\Gamma, x : A \vdash x : A}\ Assumption \qquad \frac{\Gamma \vdash M : A}{\Gamma \rhd hd(M) : A}\ Coercion$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \supset B}\ Intro \qquad \frac{\Gamma \rhd F : A \supset B \quad \Gamma \vdash N : A}{\Gamma \rhd FN : B}\ mpElim$$

$$\frac{\Gamma \rhd F : A \supset B \quad \Gamma \vdash N : A \quad \Gamma, x : B \vdash P : C}{\Gamma \vdash app(F, N, (x)P) : C}\ gmElim$$

In accordance with what was observed before, this is a natural deduction system extending that of von Plato's. The system contains two primitive elimination rules, *general multiary elimination* (or *outer elimination*) and *inner elimination*, the latter being a form of *modus ponens*. There is a further rule, whose instances are called *coercions*, with *coercion formula* A. The general elimination rule is recovered as a combination of outer elimination and coercion. In addition, a sequent of the form $\Gamma \rhd F : A$ occurs in a derivation of the system iff it occurs as the major premiss of an elimination and A is an implication. The *Coercion* rule, then, means that any sequent of the first kind can serve as major premiss of an elimination.

There are three reduction rules: two β-rules

$$\begin{array}{ll} (\beta 1) & app(hd(\lambda x.M), N, (y)P) \to [[N/x]M/y]P \\ (\beta 2) & hd(\lambda x.M)N \to hd([N/x]M)\ , \end{array}$$

and rule (π)

$$\begin{array}{l} app(hd(app(F, N, (x)P))N_1...N_{m-1}, N_m, (y)P') \to \\ \qquad \to app(F, N, (x)app(hd(P)N_1...N_{m-1}, N_m, (y)P'))\ , \end{array}$$

where $m \geq 1$. Let $\beta = \beta 1 \cup \beta 2$. By *gm-normalisation* we mean $\beta\pi$-reduction. Notice that rule $\beta 2$ is a relation on functions. As to rule π, if $F = hd(M)$ and $m = 1$, we recognize the π rule of λg. In the general case, rule π is non-local, because, if we let the redex be $app(F', N_m, (y)P)$, it requires the full inspection of F' until the head emerges. Also a $\beta 2$-reduction step requires the full inspection of function F of the application $app(F, N, (y)P)$ where the $\beta 2$-redex is located.

In λNm, a λg-terms is a term without occurrences of FN (hence every gm-application in a λg-term is a g-application $app(hd(M), N, (x)P)$). M is in $\beta\pi$-nf iff every coercion in M is of the form $hd(x)$. A derivation $\mathcal{D}$ in λNm is $\beta\pi$-normal iff *every coercion formula occurring in* $\mathcal{D}$ *is an assumption.* In particular, this gives von Plato's criterion of normality for λg-terms, because in λg coercion formula = main premiss of elimination.

Remark: Both λGm and λNm are new presentations of the system $\lambda\mathbf{J^m}$ of [3]. A gm-application is written there $t(u_1, [u_2, ..., u_n], (x)v)$. This representation brings to the surface both the head t and the "continuation" $(x)v$. The price to pay for these advantages is that the presentation in *op. cit.* has a hybrid

proof-theoretical character. The typing rule of the gm-application constructor in *op. cit.* is an elimination rule, but lists l of a restricted form are primitive. Nevertheless, λGm and λNm inherit the properties of $\lambda\mathbf{J^m}$ [4]:

Theorem 1. *In λGm and λNm, $\beta\pi$-reduction is s.n. on typable terms and confluent.*

Mappings Θ and Ψ: We now define mappings between the set of λGm-terms and the set of λNm-terms. Once and for all, variables and λ-abstractions are mapped identically. The question will always be how to map cuts, left introductions and eliminations.

We start with a mapping $\Psi : \lambda Nm{-}Terms \longrightarrow \lambda Gm{-}Terms$. Let $\Psi(M) = t, \Psi(N_i) = u_i$ and $\Psi(P) = v$. The idea is to map, say, $app(hd(M)N_1N_2, N_3, (x)P)$ to $t(u_1 :: u_2 :: u_3 \cdot (x)v)$. This is achieved with the help of an auxiliary function $\Psi : \lambda Nm - Functions \times \lambda Gm - Lists \longrightarrow \lambda Gm - Terms$ as follows:

$$\begin{aligned} \Psi(x) &= x & \Psi(hd(M), l) &= (\Psi M)l \\ \Psi(\lambda x.M) &= \lambda x.\Psi M & \Psi(FN, l) &= \Psi(F, \Psi N :: l) \\ \Psi(app(F, N, (x)P)) &= \Psi(F, \Psi N \cdot (x)\Psi P) \end{aligned}$$

Next we consider a mapping $\Theta : \lambda Gm - Terms \longrightarrow \lambda Nm - Terms$. Let $\Theta(t) = M$, $\Theta(u_i) = N_i$ and $\Theta(v) = P$. The idea is to map, say, $t(u_1 :: u_2 :: u_3 \cdot (x)v)$ to $app(hd(M)N_1N_2, N_3, (x)P)$. This is achieved with the help of an auxiliary function $\Theta : \lambda Nm - Functions \times \lambda Gm - Lists \longrightarrow \lambda Nm - Terms$ as follows:

$$\begin{aligned} \Theta(x) &= x & \Theta(F, u \cdot (x)v) &= app(F, \Theta u, (x)\Theta v) \\ \Theta(\lambda x.t) &= \lambda x.\Theta t & \Theta(F, u :: l) &= \Theta(F\Theta u, l) \\ \Theta(tl) &= \Theta(hd(\Theta t), l) \end{aligned}$$

Theorem 2 (Isomorphism). *Mappings Ψ and Θ are sound, mutually inverse bijections between the set of λGm-terms and the set of λNm-terms. Moreover, for each $R \in \{\beta 1, \beta 2, \pi\}$:*

1. *$M \to_R M'$ in λNm iff $\Psi M \to_R \Psi M'$ in λGm.*
2. *$t \to_R t'$ in λGm iff $\Theta t \to_R \Theta t'$ in λNm.*

The proof follows the pattern of proof of similar results in [2].

3 Unification

A problem of wrong focus: Applicative terms in λGm and λNm have the form of cuts and gm-eliminations

$$t(u_1 :: ... :: u_{m-1} :: u_m \cdot (x)v) \ , \tag{1}$$

$$app(hd(M)N_1...N_{m-1}, N_m, (x)P) \ . \tag{2}$$

respectively. In both cases there is a head, m arguments ($m \geq 1$) and a tail (or continuation). In the first case, the term is split next to the head, with the rest of data organized as a list l; in the second case, the term is split just before the tail, with the rest of data organized as a function F. In the first case, the head is focused, in the second it is the tail that is focused.

Now both cut-elimination and gm-normalisation aim at reducing heads to variables, and are a process of transforming heads. In this respect, the focus of tails is unfortunate and explains the fact that both $\beta 2$ and π are non-local reduction rules in the natural deduction system λNm.

Non-local rules are bad, for instance, for the implementation of λNm, where the search for heads has to be made explicitly. The solution we propose is to extend λNm to a calculus where applicative terms are split at arbitrary position, and not just around the tail. This means that both functions F and lists l are used in the representation of applicative terms, and this representation turns out to unify both cuts and gm-eliminations.

The telescopic effect: The idea of manipulating functions F and lists l in the same system has many motivations. For instance, the intuition about lists l is that they are "applicative contexts", prescribing a linear and immediate use to some expression to be supplied; symmetrically, functions F are expressions which are used in a linear and immediate way (in the function position of some application). But so far functions and lists live in separate systems.

Another motivation is as follows. Let M_0 be the gm-application (2), and let $\Theta t = M$, $\Theta u_i = N_i$ and $\Theta v = P$. There are m choices of F, l such that $M_0 = \Theta(F, l)$, ranging from the choice $F = hd(M)N_1...N_{m-1}$ and $l = u_m \cdot (x)v$ to the choice $F = hd(M)$ and $l = u_1 :: ...u_{m-1} :: u_m \cdot (x)v$. This last case is particularly important, because the representation of application M_0 as $\Theta(hd(M), l)$, for such l, brings to the surface the head $hd(M)$. In general, we will use pattern matching of gm-application with $\Theta(hd(M), l)$ to obtain the effect of extracting the head of the application, an effect we call the *telescopic* effect. Similarly one extracts the tail of cuts by pattern-matching with $\Psi(F, u \cdot (x)v)$.

The telescopic effect is useful in making global rules look local. This is achieved by manipulating simultaneously, in the meta-language, both functions F and lists l. For instance, reduction rule π in λNm may be defined as follows:

$$(\pi) \qquad \Theta(hd(app(F, N, (x)P)), l) \rightarrow app(F, N, (x)\Theta(hd(P), l)) \ .$$

The calculus we introduce next manipulates expressions $\Theta(F, l)$ formally.

The unified calculus: Expressions in λU are given by:

$$\begin{array}{lr} \text{(Terms)} & M, N, P ::= x \mid \lambda x.M \mid \underline{\theta}(F, L) \\ \text{(Functions)} & F ::= hd(M) \mid FN \\ \text{(Lists)} & L, K ::= N :: L \mid N \cdot (x)P \end{array}$$

$\underline{\theta}(F, L)$ is called a *unified cut*. The symbol $\underline{\theta}$ is a formal counterpart of Θ. In $\underline{\theta}(F, L)$, we say that F is in *focus*. The new typing rule is:

$$\frac{\Gamma \rhd F : A \quad \Gamma; L : A \vdash B}{\Gamma \vdash \underline{\theta}(F, L) : B}\ uCut$$

In λU, a *sequent term* is a term with no occurrences of FN, *i.e* an elimination-free term, whereas a *natural deduction term* is a term with no occurrences of $N :: L$, *i.e.* a left-introduction-free term. In sequent terms and natural deduction terms, unified cuts have the form

$$ML = \underline{\theta}(hd(M), L) \qquad app(F, N, (x)P) = \underline{\theta}(F, N \cdot (x)P)$$

respectively. These equations show how unified cut unifies cut and and gm-elimination. Sequent terms (resp. natural deduction terms) dispense with the syntactic class of functions F (resp. lists L) and constitute a copy of λGm-terms (resp. λNm-terms) in λU. Given a λGm-term t (resp. λNm-term M), we denote by t' (resp. M') its copy in λU.

The reduction rules of λU are as follows:

$$\begin{array}{rr}
(\beta 1) & \underline{\theta}(hd(\lambda x.M), N \cdot (y)P) \to [[N/x]M/y]P \\
(\beta 2) & \underline{\theta}(hd(\lambda x.M), N :: L) \to \underline{\theta}(hd([N/x]M), L) \\
(\pi) & \underline{\theta}(hd(\underline{\theta}(F, L)), K) \to \underline{\theta}(F, L@K) \\
(\underline{\psi}) & \underline{\theta}(FN, L) \to \underline{\theta}(F, N :: L)
\end{array}$$

Let $\beta = \beta 1 \cup \beta 2$. Rules β and π require a head in focus. For this reason, are local transformations. Rule $\underline{\psi}$ is a step towards focusing a head. A λU-term is a $\underline{\psi}$-nf iff it is a sequent term. $\underline{\psi}$-reduction is terminating (it decreases the number of occurrences of FN) and locally confluent. Hence it is confluent. We denote by $\underline{\psi}(M)$ the unique $\underline{\psi}$-nf of a λU-term M. It holds, for all $M \in \lambda Nm$, that

$$\underline{\psi}(M') = \Psi(M)' \ .$$

It is easy to see that sequent terms are closed for $\beta\pi$-reduction, and a λGm-term t $\beta\pi$-reduces in λGm exactly as t' $\beta\pi$-reduces in λU. Let us see what happens when we $\beta\pi$-reduce M' in λU, for $M \in \lambda Nm$.

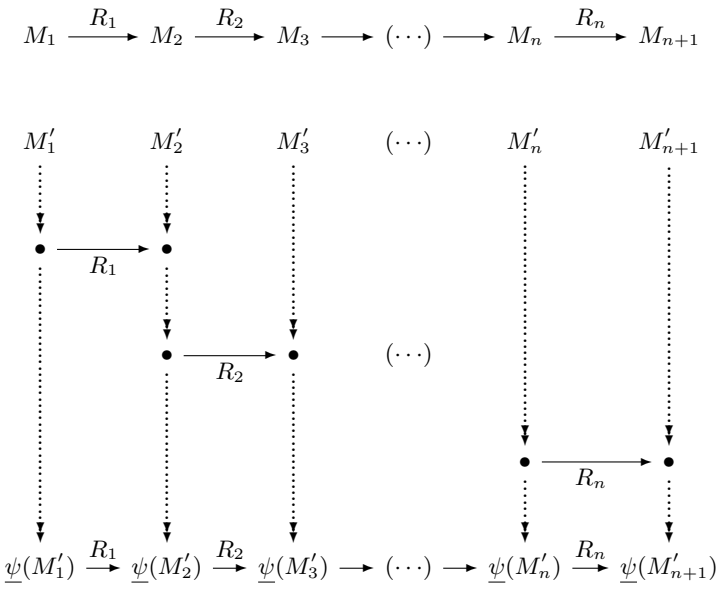
Proposition 1. *Let $R \in \{\beta 1, \beta 2, \pi\}$.*

1. *In λU, if $M \to_R M_1$ and $M \to_{\underline{\psi}} M_2$ then there is M_3 such that $M_2 \to_R M_3$ and $M_1 \to^*_{\underline{\psi}} M_3$.*
2. *If $M \to_R N$ in λNm, then there are M_1, N_1 such that, in λU: $M_1 \to_R N_1$ and $M \to^*_{\underline{\psi}} M_1$ and $N \to^*_{\underline{\psi}} N_1$.*

Theorem 3. *Suppose $M_1 \to_{R_1} M_2 \to (\cdots) \to M_n \to_{R_n} M_{n+1}$ is a $\beta\pi$-reduction sequence in λNm (hence each $R_i \in \{\beta 1, \beta 2, \pi\}$). Then, the reductions in λU depicted in Fig. 1 hold, when vertical arrows denote $\underline{\psi}$-reduction.*

Proof: By induction on n, using the previous proposition and confluence of $\underline{\psi}$. ■

Fig. 1. Normalisation in λU



Regarding Fig. 1 again, we can now compare reduction of M_1 in λNm with reduction of M'_1 in λU. The latter is obtained from the former by interleaving $\underline{\psi}$-reduction steps. To a possibly non-local reduction step $\rightarrow_{R_i}$ in the former corresponds a necessarily local reduction step $\rightarrow_{R_i}$ in the latter. The interleaved $\underline{\psi}$-reduction steps do explicitly the focusing of heads implicit in the reduction steps at the λNm level. The reduction of $\underline{\psi}(M'_1)$ is morally the same as the reduction of $\Psi(M_1)$ in λGm. Fig. 1 is a refinement of the "only if" part of statement 1 in Theorem 2.

Finally, observe that part 1 of Proposition 1 allows the projection of $\beta\pi\underline{\psi}$-reduction sequences of λU into $\beta\pi$-reduction sequences of λGm. So, it is easy to lift Theorem 1 from λGm to λU.

Theorem 4. *$\beta\pi\underline{\psi}$-reduction in λU is s.n. on typable terms and confluent.*

Variant of the unified calculus: Consider a variant of permutative conversion p of $\lambda\mathbf{J^m}$ [3], given here for λGm with the help of telescopic effect:

$$(p) \qquad \Psi(F, u \cdot (x)v) \rightarrow [\Psi(F,[u])/x]v, \text{ if } v \neq x \ .$$

This rule acts on tails, eliminating occurrences of general left-introduction. It is a non-local rule in λGm. A variant of the unified calculus, seen as an extension of λGm, can be defined for tail-active conversions, with terms:

$$t, u, v \ ::= \ x \,|\, \lambda x.t \,|\, \underline{\psi}(f, l)$$

The p-rule now reads $\underline{\psi}(f, u \cdot (x)v) \to [\underline{\psi}(f,[u])/x]v$. In $\underline{\psi}(f,l)$ the focus is l and a rule $\underline{\theta}$ is needed for bringing continuations to focus: $\underline{\psi}(f, u :: l) \to \underline{\psi}(fu, l)$.

4 Conclusions

From a logical point of view, λU achieves the same goal as the "uniform" calculus of [8], but with a radically different approach (the latter approach is to extend natural deduction with general elimination and general introduction rules).

It is to be expected that λU admits extensions (encompassing a sequent calculus where cuts are not necessarily right-principal) and further variants. For instance, consider the following rules for λNm, given with telescopic effect:

$$(\mu) \qquad \Theta(F, N \cdot (x)\Theta(hd(x), l)) \to \Theta(FN, l), \text{ if } x \notin l \ .$$

This is a natural deduction variant of rule μ introduced in [9]. Consider the μ-redex. We analyze the tail of the outer applicative term and the head of the inner applicative term. This rules needs a mix of head and tail focus. Maybe a good system for dealing with such rules is a variant of the unified calculus with reduction modulo the equation $\underline{\theta}(FN, L) = \underline{\theta}(F, N :: L)$.[1]

References

1. R. Dyckhoff and L. Pinto. Permutability of proofs in intuitionistic sequent calculi. *Theoretical Computer Science*, 212:141–155, 1999.
2. J. Espírito Santo. *Conservative extensions of the λ-calculus for the computational interpretation of sequent calculus.* PhD thesis, University of Edinburgh, 2002. Available at http://www.lfcs.informatics.ed.ac.uk/reports/.
3. J. Espírito Santo and Luís Pinto. Permutative conversions in intuitionistic multiary sequent calculus with cuts. In M. Hoffman, editor, *Proc. of TLCA'03*, volume 2701 of *Lecture Notes in Computer Science*, pages 286–300. Springer-Verlag, 2003.
4. J. Espírito Santo and Luís Pinto. Confluence and strong normalisation of the generalised multiary λ-calculus. In Ferruccio Damiani Stefano Berardi, Mario Coppo, editor, *Revised selected papers from the International Workshop TYPES 2003*, volume 3085 of *Lecture Notes in Computer Science*. Springer-Verlag, 2004.
5. 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.
6. F. Joachimski and R. Matthes. Short proofs of normalization for the simply-typed lambda-calculus, permutative conversions and Gödel's T. *Archive for Mathematical Logic*, 42:59–87, 2003.
7. G. Mints. Normal forms for sequent derivations. In P. Odifreddi, editor, *Kreiseliana*, pages 469–492. A. K. Peters, Wellesley, Massachusetts, 1996.
8. S. Negri and J. von Plato. *Structural Proof Theory.* Cambridge, 2001.
9. H. Schwichtenberg. Termination of permutative conversions in intuitionistic gentzen calculi. *Theoretical Computer Science*, 212, 1999.
10. J. von Plato. Natural deduction with general elimination rules. *Annals of Mathematical Logic*, 40(7):541–567, 2001.

[1] Acknowledgment: The diagram in Fig. 1 was produced with Paul Taylor's macros.