The λ-calculus and the unity of structural proof theory

José Espírito Santo

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

Abstract. In the context of intuitionistic implicational logic, we achieve a perfect correspondence (technically an isomorphism) between sequent calculus and natural deduction, based on perfect correspondences between left-introduction and elimination, cut and substitution, and cut-elimination and normalisation. This requires an enlarged system of natural deduction that refines von Plato's calculus. It is a calculus with *modus ponens* and primitive substitution; it is also a "coercion calculus", in the sense of Cervesato and Pfenning. Both sequent calculus and natural deduction are presented as typing systems for appropriate extensions of the λ-calculus. The whole difference between the two calculi is reduced to the associativity of applicative terms (sequent calculus = right associative, natural deduction = left associative), and in fact the achieved isomorphism may be described as the mere inversion of that associativity.
The novel natural deduction system is a "multiary" calculus, because "applicative terms" may exhibit a list of several arguments. But the combination of "multiarity" and left-associativity seems simply wrong, leading necessarily to non-local reduction rules (reason: nomalisation, like cut-elimination, acts at the *head* of applicative terms, but natural deduction focuses at the *tail* of such terms). A solution is to extend natural deduction even further to a calculus that unifies sequent calculus and natural deduction, based on the unification of cut and substitution. 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.

1 Introduction

Two of the most important systems of formal deduction are sequent calculus and natural deduction, both introduced in Gentzen's seminal paper [15]. When they were introduced, the two systems seemed to differ substantially. Natural deduction manipulated formulas, tried to model informal reasoning, and had an implicit management of structural rules. Sequent calculus manipulated "sequents" (formal instances of the provability relation), tried to model a symmetry between assumption and conclusion, and had explicit management of structural rules; in addition, it enjoyed a *hauptsatz*, the cut-elimination theorem. Other differences were realized over the years. The cut-free fragment of sequent calculus is appropriate for meta-theoretical reasoning and proof search, and therefore has a link with logic programming [14, 20]. Natural deduction, via the Curry-Howard isomorphism [3, 17], has a link with λ-calculus and functional programming.

One of the main tasks of structural proof theory [22] is to investigate whether these differences between sequent calculus and natural deduction (and the concomitant relative (dis)advantages and different applications of the systems) are absolute or just apparent. This task has been carried out over the last 70 years [15, 25, 28, 24, 21, 27], and the outcome is that the differences are most of the time just apparent. Gentzen observed already in [15] that natural deductions can be seen as trees of sequents. Kleene [19] showed how to define sequent calculus with implicit structural rules, built in the logical rules. Prawitz [25] showed that natural deduction also enjoys a *hauptsatz*, the normalisation theorem, from which many meta-theoretical results can be obtained easily. Zucker and Pottinger [28, 24] showed that there is a "homomorphism" between the process of cut-elimination and the process of normalisation. von Plato [27] enlarged the concept of natural deduction so that cut-free derivation and normal natural deduction are in bijective correspondence. Finally, Herbelin [16] started a definitive extension of the Curry-Howard isomorphism to the sequent calculus.

In this paper we continue the contributions of von Plato and Herbelin, in the context of intuitionistic implicational logic. We adopt the Curry-Howard approach of defining logical systems as typing systems for some variant of the λ-calculus, both for sequent calculus (following Herbelin) and natural deduction. This has some technical advantages (uniform treatment of structural rules, handling of proofs as terms) and brings in computational meaningful term language for interpreting the proofs and expressing the differences of the systems. We also adopt (and are ready to refine) von Plato's extension of natural deduction. The outcome of the paper is a new level of understanding of the correspondence between sequent calculus and natural deduction, leading to a *two-staged* comprehension of the unity of structural proof theory: first, sequent calculus and natural deduction are *isomorphic* systems which differ in a single feature, the *associativity of applicative terms*; second, sequent calculus and natural deduction coexist inside a larger system, the *unified calculus*, based on the *unification* of cut and substitution. We now explain these two stages in some detail.

Isomorphism stage: Herbelin studied a fragment LJT of sequent calculus LJ and gave its computational interpretation in terms of the so-called $\overline{\lambda}$-calculus. Contrary to earlier accounts of the computational interpretation of the sequent calculus, whose focus was on the feature of pattern matching, in $\overline{\lambda}$ the novelty is the existence of an auxiliary syntactic class of *applicative contexts*. In the case of intuitionistic implication, an applicative context is simply a list of terms, understood as a "multiary" argument for functional application. Hence, "applicative terms" in $\overline{\lambda}$ have the form $t[u_1, ..., u_m]$. Herbelin expressed the difference between sequent calculus and natural deduction as the difference between $t[u_1, ..., u_m]$ and the ordinary $MN_1...N_m$ of the λ-calculus.

There is a subtle point here. The difference between $t[u_1, ..., u_m]$ and $MN_1...N_m$ includes a difference in the organization of applicative terms: sequent calculus is right-associative $t(u_1 :: ...(u_m :: []))$, whereas natural deduction is left-associative $(...(MN_1)...N_m)$. But the difference between $t[u_1, ..., u_m]$ and $MN_1...N_m$ *cannot* be reduced to an inversion of the associativity of applicative terms. That is true only in the cut-free and normal fragments: the mapping that inverts the associativity of applicative terms is proved in [4] to be a bijection between the structures $x[u_1, ..., u_m]$ and $xN_1...N_m$. But in the unconstrained case, $t[u_1, ..., u_m]$ is more general than $MN_1...N_m$. For instance, $t[u_1, u_2][u_3]$ and $t[u_1, u_2, u_3]$ both correspond to $MN_1N_2N_3$, because, in the λ-calculus, we do not have a way of delimiting the applicative term MN_1N_2 and saying it is the head of a new applicative term. Indeed, the λ-calculus is isomorphic (even at level of reduction) to the structure $V[u_1, ..., u_m]$ (where V is a value, *i.e.* a variable or abstraction) [6, 7], and only an extension of the λ-calculus, equipped with a distinction between applicative term and application, is isomorphic to the structure $t[u_1, ..., u_m]$ [7, 8].

These developments may be seen as the early steps of a programme, named *Herbelin's programme* in [9]. The programme consists in investigating whether, for increasingly larger fragments of sequent calculus, there are isomorphic extensions of natural deduction, so that the sequent calculus fragments and the natural deduction extensions are linked by a mere inversion of applicative terms. The benefit of the programme for sequent calculus is that only an isomorphic natural deduction system gives rigorous meaning to "applicative context" and "applicative term", and for natural deduction is that the extent of the natural deduction "space of calculi" is uncovered.

The developments in [6–8] only comprise, in the sequent calculus side, fragments of LJT. But LJT is a permutation-free fragment, where only a restricted form of left introduction is available and where the computational meaning of permutation (so typical of sequent calculus) is absent. So the challenge, taken in the "isomorphism stage", is to complete Herbelin's programme for full sequent calculus, that is, sequent calculus without constraints on left introductions (but where permutative conversions necessarily show up). The computational interpretation will be in terms of a λ-calculus λ^{Gtz} (named so after Gentzen) with a primitive notion of applicative context, the latter taken in a certain generalised sense. In the natural deduction side, a system λ_{Nat} is defined that extends and refines von Plato's natural deduction. It is a calculus with *modus ponens* and primitive substitution, and integrates the idea of a distinction between applicative term and application. Such syntactic structure turns out to expand the idea of defining natural deduction as a "coercion calculus", in the sense of Cervesato and Pfenning [1].

Then we prove $\lambda^{\mathsf{Gtz}} \cong \lambda_{\mathsf{Nat}}$ in the fullest sense: the mapping that inverts the associativity of applicative terms is a sound bijection between the sets of terms of the two calculi and, in addition, establishes an isomorphism between cut-elimination in λ^{Gtz} and normalisation in λ_{Nat}. Strong cut-elimination for λ^{Gtz} is proved via an interpretation into the calculus of "delayed substitutions" $\lambda\mathsf{s}$ of [10]; strong normalisation for λ_{Nat} follows by isomorphism. These results constitute, for the logic under analysis here, considerable improvements over [16, 1, 27, 8].

Unification stage: After pointing out so accurately the single feature that distinguishes sequent calculus from natural deduction, one expects to obtain a transparent view of the relative virtues and defects of the two systems. We will fulfill this expectation, and even suggest a remedy to the defects of each system.

The notion of applicative term consists, both in λ^{Gtz} and λ_{Nat}, of a head term, m arguments ($m \geq 0$), and a "continuation" or tail. Let us call "multiarity" the possibility of forming applicative terms with $m > 1$ arguments. Although normalisation in λ_{Nat} is isomorphic to cut-elimination in λ^{Gtz}, the combination of multiarity and left-associativity of applicative terms in λ_{Nat} implies that 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 the natural deduction representation of applicative terms focuses at the *tail* of such terms. Now, in a multiary system, heads are arbitrarily distant from tails. So, we get isomorphism, but normalisation 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 suggested by the analysis of $\lambda^{\mathsf{Gtz}} \cong \lambda_{\mathsf{Nat}}$, is to extend natural deduction even further, to a calculus $\lambda\mathsf{U}$ that *unifies* λ^{Gtz} and λ_{Nat}. This *unified calculus* is based on the unification of cut and substitution, and all its reduction rules are local. A sequent term behaves in $\lambda\mathsf{U}$ like in the sequent calculus, whereas the reduction steps in $\lambda\mathsf{U}$ 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 which 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.

Remark: This paper is based on [9, 11]. Both papers prove isomorphisms that go beyond Herbelin's fragment LJT: $\lambda Gm \cong \lambda Nm$ in [11] and $\lambda^{\mathsf{Gtz}} \cong \lambda_{\mathsf{Nat}}$ in [9]. λGm is a fragment of λ^{Gtz} where cuts have to be "right principal", and λNm is a fragment of λ_{Nat} where substitutions have necessarily a form that corresponds to generalised application. There is a uniform explanation of these restriction in λ-calculus terms: in applicative terms, $m \geq 1$. These fragments, as well as $\lambda Gm \cong \lambda Nm$, are omitted here, so we only present the isomorphism originally proven in [9]. On the other hand, the unified calculus is studied only in [11], with name λU. This calculus, which unifies λGm and λNm, based on the unification of right-principal cut and generalised application, is also omitted here. The reason is that we introduce here a new, slightly more general, unified calculus $\lambda\mathsf{U}$, based on the unification of cut and substitution, unifying λ^{Gtz} and λ_{Nat}. See Fig. 1 for a road-map, where Θ and Ψ denote the (inverse) isomorphisms.

Fig. 1. Completing Herbelin's programme, towards a unification

$$\begin{tikzcd}
 & \lambda\mathsf{U} & \\
\lambda^{\mathsf{Gtz}} \arrow[ur] \arrow[rr, shift left, "\Theta"] & & \lambda_{\mathsf{Nat}} \arrow[ul] \arrow[ll, shift left, "\Psi"] \\
 & \lambda U \arrow[uu] & \\
\lambda Gm \arrow[uu] \arrow[ur] \arrow[rr, shift left, "\Theta"] & & \lambda Nm \arrow[uu] \arrow[ul] \arrow[ll, shift left, "\Psi"]
\end{tikzcd}$$

Structure of the paper: The paper is organized as follows. Section 2 presents λ^{Gtz}. Section 3 presents λ_{Nat}. Sections 4 and 5 prove and analyze $\lambda^{\mathsf{Gtz}} \cong \lambda_{\mathsf{Nat}}$. Section 6 motivates, defines and studies the unified calculus. Section 7 discusses the contributions of the paper and related work. Section 8 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 produces a consistent set. Meta-substitution is denoted with square brackets $[\_/x]\_$. All calculi in this paper assume Barendregt's variable convention (in particular we take renaming of bound variables for granted).

Naming of systems: sequent calculi are denoted λ^S (where S is some tag); natural deduction systems are denoted λ_S. There are two exceptions, borrowed from [11]: λGm (a sequent calculus) and λNm (a natural deduction system). Other system which we will regard as being in the intersection of, or which unify, sequent calculus and natural deduction are denoted λS.

2 Sequent calculus

The sequent calculus we introduce is named λ^{Gtz} (read "λ-Gentzen").

Expressions and typing rules: There are two sorts of expressions in λ^{Gtz}: terms t, u, v and contexts k.

$$\begin{array}{lr}
\text{(Terms)} & t, u, v ::= x \,|\, \lambda x.t \,|\, tk \\
\text{(Contexts)} & k ::= (x)v \,|\, u :: k
\end{array}$$

Terms are either variables x, y, z, abstractions $\lambda x.t$ or *cuts* tk. Contexts are either a *selection* $(x)v$ or a *linear left introduction* $u :: k$, often called a *cons*. Variable

x is bound in $(x)v$. [1] A computational reading of contexts is as a prescription of what to do next (with some expression that has to be plugged in). A selection $(x)v$ says "substitute for x in v" and a cons $u :: k$ says "apply to u and proceed according to k". A cut tk is a plugging of a term t in the context k. We will use the following abbreviations:

$$\begin{aligned} [] &:= (x)x \\ [u_1, ..., u_n] &:= u_1 :: ...u_n :: [] \\ \langle u/x \rangle t &:= u(x)t \end{aligned}$$

The typing rules of λ^{Gtz} are as follows:

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

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

$$\frac{\Gamma \vdash t : A \quad \Gamma; A \vdash k : B}{\Gamma \vdash tk : B}\ Cut \qquad \frac{\Gamma, x : A \vdash v : B}{\Gamma; A \vdash (x)v : B}\ Selection$$

There are two sorts of sequents in λ^{Gtz}, namely $\Gamma \vdash t : A$ and $\Gamma; A \vdash k : B$. The distinguished position in the antecedent of sequents of the latter kind contains the *selected* formula. There is a typing rule *Selection* that selects an antecedent formula. Besides this rule, there are the axiom rule, the introductions on the left(=antecedent) and on the right(=succedent) of sequents, and the cut.

The typing rules follow a reasonable discipline: active formulas in the antecedent of sequents have to be previously selected (the B in $Left$ and one A in Cut); and a formula introduced on the left is selected. The latter constraint implies that a left introduction $u :: k$ is a *linear* introduction, because there cannot be an implicit contraction. Full left introduction is recovered as a cut between an axiom and a linear left introduction, corresponding to $x(u :: k)$. The cut-elimination process will not touch these trivial cuts. More generally, given a context k, xk represents the inverse of a selection, that is, the operation that takes a formula out of the selection position and gives it name x.[2] An implicit contraction may happen here.

Reduction rules: The reduction rules of λ^{Gtz} are as follows:

$$\begin{array}{llll} (\beta) & (\lambda x.t)(u :: k) \to \langle u/x \rangle (tk) & (\sigma) & \langle t/x \rangle v \to [t/x]v \\ (\pi) & (tk)k' \to t(k@k') & (\mu) & (x)xk \to k, \text{ if } x \notin k \end{array}$$

where

[1] In order to save parentheses, the scope of binders extends to the right as far as possible.

[2] Such inference rule is primitive in Herbelin's *LJT* and called "dereliction" in [16] and "selection" in [4]. The latter name, of course, comes from a bottom-up reading.

$$(u :: k)@k' = u :: (k@k') \qquad ((x)v)@k' = (x)vk'$$

By *cut-elimination* we mean $\beta\pi\sigma$-reduction. Rules β, π and σ aim at eliminating all cuts that are not of the form $x(u :: k)$. The procedure is standard. If a cut is a key-cut (both cut-formulas main(=introduced) in the premisses) with cut-formula $A \supset B$, the cut is reduced to two cuts, with cut-formulas A and B. This is rule β. If a cut, not of the form $x(u :: k)$, is not a key cut, this means that it can be permuted to the right (rule σ) or to the left (rule π). The particular case of σ when $v = x$ is named ϵ and reads $\langle t/x\rangle x \to t$ or $t[] \to t$. A term t is a $\beta\pi\sigma$-normal form iff it is generated by the following grammar:

$$\begin{array}{c} t, u, v ::= x \mid \lambda x.t \mid x(u :: k) \\ k ::= (x)v \mid u :: k \end{array} \tag{1}$$

There is a further reduction rule, named μ, of a different nature. It undoes the sequence of inferences consisting of un-selecting and selecting the same formula, if no implicit contraction is involved. A similar rule has been defined for Parigot's $\lambda\mu$-calculus [23], but acting on the RHS of sequents.

Consider the term $(\lambda x.t)(u :: k)$. After a β-step, we get $v = \langle u/x\rangle(tk)$. If u is a cut $t'k'$, v is both a σ- and a π-redex. In this case, there is a choice as to how to continue evaluation. Opting for σ gives $([u/x]t)k$, whereas the π option gives $t'(k'@(x)tk)$. According to [2], this choice is a choice between a call-by-name and a call-by-value strategy of evaluation.

Strong normalisation: We give a proof of strong normalisation for λ^{Gtz} by defining a reduction-preserving interpretation in the $\lambda\mathsf{s}$-calculus of [10].

The terms of $\lambda\mathsf{s}$ are given by:

$$M, N, P ::= x \mid \lambda x.M \mid MN \mid \langle N/x\rangle M$$

This set of terms is equipped with the following reduction rules:

$$\begin{array}{llll} (\beta) & (\lambda x.M)N \to \langle N/x\rangle M & (\pi_1) & (\langle P/x\rangle M)N \to \langle P/x\rangle(MN) \\ (\sigma) & \langle N/x\rangle M \to [N/x]M & (\pi_2) & \langle\langle P/y\rangle N/x\rangle M \to \langle P/y\rangle\langle N/x\rangle M \end{array}$$

where meta-substitution $[N/x]M$ is defined as expected. In particular

$$[N/x]\langle P/y\rangle M = \langle [N/x]P/y\rangle [N/x]M \ .$$

Let $\pi = \pi_1 \cup \pi_2$. We now define a mapping $(\_)^* : \lambda^{\mathsf{Gtz}} \to \lambda\mathsf{s}$. More precisely, mappings $(\_)^* : \lambda^{\mathsf{Gtz}} - Terms \to \lambda\mathsf{s} - Terms$ and $(\_,\_)^* : \lambda\mathsf{s} - Terms \times \lambda^{\mathsf{Gtz}} - Contexts \to \lambda\mathsf{s} - Terms$ are defined by simultaneous recursion as follows:

$$\begin{array}{rlrl} x^* &= x & (M, (x)v)^* &= \langle M/x\rangle v^* \\ (\lambda x.t)^* &= \lambda x.t^* & (M, u :: k)^* &= (Mu^*, k)^* \\ (tk)^* &= (t^*, k)^* & & \end{array}$$

The idea is that, if t, u_i and v are mapped by $(\_)^*$ to M, N_i and P, respectively, then $t(u_1 :: \cdots u_m :: (x)v)$ is mapped to $\langle MN_1 \cdots N_m/x\rangle P$.

Proposition 1. *Let $R \in \{\beta, \pi, \sigma, \mu\}$. If $t \to_R u$ in λ^{Gtz}, then $t^* \to^+_{\beta\pi\sigma} u^*$ in $\lambda\mathsf{s}$.*

Proof: Follows from the following four facts: (i) $(\langle N/x\rangle M, k)^* \to^+_\pi \langle N/x\rangle (M,k)^*$; (ii) $((M,k)^*, k')^* \to^+_\pi (M, k@k')^*$; (iii) $([t/x]u)^* = [t^*/x]u^*$; and (iv) $\langle M/x\rangle (N,k)^* \to_\sigma ([M/x]N, k)^*$, if $x \notin k$. ■

Theorem 1 (Strong cut-elim.). *Every typable $t \in \lambda^{\mathsf{Gtz}}$ is $\beta\pi\sigma\mu$-SN.*

Proof: [10] proves that every typable $t \in \lambda\mathsf{s}$ is $\beta\pi\sigma$-SN (if we use for $\lambda\mathsf{s}$ the obvious typing rules). The theorem follows from this fact, the previous proposition and the fact that $(\_)^*$ preserves typability. ■

Related systems: We can easily embed LJ in λ^{Gtz}, if we define LJ as the typing system for some obvious term language. The embedding is given by:

$$\begin{array}{rcl} \mathsf{Axiom}(x) \rightsquigarrow x & \qquad & \mathsf{Left}(x, L_1, (y)L_2) \rightsquigarrow x(u_1 :: (y)u_2) \\ \mathsf{Right}((x)L) \rightsquigarrow \lambda x.t & \qquad & \mathsf{Cut}(L_1, (x)L_2) \rightsquigarrow t_1(x)t_2 \end{array}$$

The cut-free LJ terms correspond to the sub-class of terms in (1) such that k in $x(u :: k)$ has to be a selection $(y)v$. $\beta\pi\sigma$-normal forms correspond thus to a generalisation of cut-free LJ-terms, namely Schwichtenberg's *multiary* cut-free terms [26]. We refer to these as *Schwichtenberg normal forms.*

A context $u_1 :: ... :: u_m :: (x)x$ ($m \geq 0$) may be regarded as a list $[u_1, ..., u_m]$, if we think of $(x)x$ as the empty list $[]$. Herbelin observes that such lists correspond to "applicative" contexts in the λ-calculus, that is, expressions of the form $[\,]N_1 \cdots N_m$, with one "hole" in the head position [16]. We will return to this issue below. If every context in a term t is of the form $[u_1, ..., u_m]$, t is a $\overline{\lambda}$-term. So, the class of λ^{Gtz}-terms generalised a sub-class of the $\overline{\lambda}$-terms. The generalisation comes from allowing selection $(x)v$ instead of just $[] = (x)x$. A term t is $\beta\pi\sigma$-normal and only contains contexts of the form $[u_1, ..., u_m]$ iff t is a cut-free $\overline{\lambda}$-term, in the sense of [16]. We refer to such terms as *Herbelin normal forms.* They are given by

$$\begin{array}{rl} t, u & ::= x \mid \lambda x.t \mid x(u :: k) \\ k & ::= [] \mid u :: k \end{array}$$

Another characterisation of this set is as the set of Schwichtenberg's normal forms (given by grammar (1)) which, in addition, are normal w.r.t. certain permutative conversions [26].

Every cut in λ^{Gtz} is of the form $t(u_1 :: ... :: u_m :: (x)v)$, with $m \geq 0$. Several interesting fragments of λ^{Gtz} may be obtained by placing restrictions on m - see Fig. 2. There is a $m \geq 1$-fragment, which gives system λGm of [11]. [3] We describe next another important fragment.

The $m \leq 1$-fragment: This fragment gives a version λ^{gs} of the λgs-calculus, to be defined in the next section. The $m \leq 1$-fragment is closed under, and equipped with, the reduction rules

[3] A notational variant of λGm, named λJ^m, is studied in [13].

Fig. 2. Fragments of λ^{Gtz} and λ_{Nat}

$$\begin{tikzcd}
\lambda^{\mathsf{Gtz}} \arrow[d, "m \geq 1"'] \arrow[dr, "m \leq 1"] & & & \lambda_{\mathsf{Nat}} \arrow[dl, "m \leq 1"'] \arrow[d, "m \geq 1"] \\
\lambda Gm \arrow[dr, "m \leq 1"'] & \lambda^{gs} \arrow[d, "m \geq 1"] & \lambda_{gs} \arrow[d, "m \geq 1"'] & \lambda Nm \arrow[dl, "m \leq 1"] \\
 & \lambda^{g} & \lambda_{g} &
\end{tikzcd}$$

$$\begin{array}{lc}
(\beta) & (\lambda x.t)(u :: (y)v) \rightarrow \langle u/x\rangle\langle t/y\rangle v \\
(\pi) & t(u :: (x)v)(u' :: (y)v') \rightarrow t(u :: (x)v(u' :: (y)v')) \\
(\sigma) & \langle t/x\rangle v \rightarrow [t/x]v
\end{array}$$

We might have considered a term like $t((x)v)(u' :: (y)v')$ as a π-redex, but we did not, for simplicity. Such a term contains a σ-redex, which must be reduced first (so the fragment is given in its call-by-name version). The fragment is also closed under some particular cases of μ-reduction (*e.g.* $t(u :: (x)x(y)v) \rightarrow t(u :: (y)v)$, with $x \notin v$), but these reductions are also σ-reductions, and we consider them as such. For instance, $t(u :: (x)x(y)v) \rightarrow_\sigma t(u :: (x)[x/y]v)$, and the latter is the same as (α-equivalent to) $t(u :: (y)v)$.

The $m \leq 1$-terms are the terms normal w.r.t. the following *permutation rule*

$$(\nu) \qquad t(u :: v :: k) \rightarrow t(u :: (z)z(v :: k)) \ ,$$

with $z \notin v, k$. Notice that $\rightarrow_\nu \subseteq \rightarrow_\mu^{-1}$. Clearly, ν is terminating (the number of ν-redexes decreases at each step). Also local confluence is easy to check. The ν-nf of t is written $\nu(t)$.

Combining restrictions $m \leq$ and $m \geq 1$, we arrive at the $m = 1$-fragment, where cuts have the form $t(u :: (x)v)$, with exactly one argument u. We denote by λ^g the class of λ^{Gtz}-terms determined by such constraint on cuts, equipped with reduction rules π (unchanged relatively to λ^{gs}) and

$$(\beta) \qquad (\lambda x.t)(u :: (y)v) \rightarrow [[u/x]t/y]v \ .$$

Substitutions $\langle u/x\rangle v$ cannot be formed, so reduction rule σ is not needed *per se*, but is implicitly employed in the new version of β. λ^g is a version of the system λg to be defined in the next section.

3 Natural deduction

The natural deduction system we introduce is named λ_{Nat} (read "λ-natural"). It is an improvement of natural deduction with general elimination rules.

Natural deduction with general elimination rules: This system [27] may be presented as a type system for the λ-calculus with generalized application. The latter is the system ΛJ of [18], 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 generalized 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:

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

Rule π corresponds to the permutative conversion allowed by general eliminations. 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 the *gElim* where $B = C$ and the rightmost premiss is omitted.

The λgs-calculus is the following version of λg with *explicit* substitution. A new term constructor, explicit substitution $\langle N/x\rangle M$, is added. In rule β

$$(\beta) \qquad (\lambda x.M)(N, y.P) \to \langle N/x\rangle\langle M/y\rangle P \ ,$$

two explicit substitutions are generated, instead of two calls to the meta-substitution. π stays the same. Finally, the calculus contains a new reduction rule, named σ, and defined by $\langle N/x\rangle M \to [N/x]M$. A λg-term is a "pure" λgs-term, that is, a λgs-term without occurrences of explicit substitution. A β-reduction step of λg can be simulated in λgs by a β-step followed by two σ-steps.[4]

Now for normal forms. In λg, the $\beta\pi$-normal terms are given by

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

and correspond to von Plato's "fully normal" natural deductions. We will refer to these as *von Plato normal forms.* A λgs-term is in $\beta\pi\sigma$-normal form iff it is a λg-term in $\beta\pi$-normal form. The class of von Plato normal forms is in bijective correspondence with cut-free LJ-terms, and, therefore, is bigger than the class of β-normal λ-terms. A $\beta\pi$-normal λg-term M is called a *Mints normal form* if, for every application $x(N, y.P)$ in M, P is y-normal [5]. P is *y-normal* if $P = y$ or $P = y(N', y'.P')$ and $y \notin N', P'$ and P' is y'-normal. Another characterisation

[4] For simplicity, we presented here the call-by-name version of λgs. For the general definition of λgs see [10].

of Mints normal forms is as $\beta\pi$-normal forms which are, in addition, normal w.r.t. a set of permutation rules given in [5]. The set of β-normal λ-terms is in bijective correspondence with the set of Mints normal forms [21, 5].

Motivation for λ_{Nat}: If one sees generalised application $M(N, x.P)$ as a substitution $\mathit{subst}(MN, x.P)$ (the notation here is not important), then one can say that in λg every ordinary application MN occurs as the actual parameter of a substitution. This situation has a defect: it is cumbersome to write iterated, ordinary applications. For instance, MNN' is written $\mathit{subst}(\mathit{subst}(MN, x.x)N', y.y)$, with x, y fresh. A solution is to allow $m \geq 0$ application as actual parameters of substitutions: $\mathit{subst}(MN_1...N_m, x.P)$. The particular case $m = 0$ encompasses explicit substitution. The usefulness of allowing $m > 1$ is precisely in having the alternative way $\mathit{subst}(MNN', x.x)$ of writing MNN'. The need to convert between the cumbersome way and the alternative way of writing MNN' leads to a new reduction rule named μ.

Expressions and typing rules: There are two syntactic classes in λ_{Nat}: terms M, N, P and *elimination expressions* E.

$$\begin{array}{ll} \text{(Terms)} & M, N, P ::= x \mid \lambda x.M \mid \{E/x\}P \\ \text{(Elimination-Expressions)} & E ::= \mathit{hd}(M) \mid EN \end{array}$$

Terms are either variables x, y, z, abstractions $\lambda x.M$ or *(primitive) substitutions* $\{E/x\}P$. Elimination expressions (EEs, for short) are either *coercions* $\mathit{hd}(M)$ (a.k.a. *heads*) or eliminations EN. So an EE is a sequence of zero or more eliminations starting from a coerced term and ending as the *actual parameter* of a substitution. Hence, every substitution has the form $\{\mathit{hd}(M)N_1...N_m/x\}P$, with $m \geq 0$. Generalised elimination is recovered as $\{\mathit{hd}(M)N/x\}P$, that is the particular case $m = 1$. Ordinary elimination is $\{\mathit{hd}(M)N/x\}x$. We will use the following abbreviations:

$$\begin{aligned} \mathit{ap}(E) &:= \{E/z\}z \\ MN &:= \mathit{ap}(\mathit{hd}(M)N) \\ M(N, y.P) &:= \{\mathit{hd}(M)N/y\}P \\ \langle N/x\rangle M &:= \{\mathit{hd}(N)/x\}M \end{aligned}$$

The typing rules of λ_{Nat} are as follows:

$$\frac{}{\Gamma, x : A \vdash x : A}\ \mathit{Assumption}$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \supset B}\ \mathit{Intro} \qquad \frac{\Gamma \rhd E : A \supset B \quad \Gamma \vdash N : A}{\Gamma \rhd EN : B}\ \mathit{Elim}$$

$$\frac{\Gamma \rhd E : A \quad \Gamma, x : A \vdash P : B}{\Gamma \vdash \{E/x\}P : B}\ \mathit{Subst} \qquad \frac{\Gamma \vdash M : A}{\Gamma \rhd \mathit{hd}(M) : A}\ \mathit{Coercion}$$

There are two sorts of sequents in λ_{Nat}, namely $\Gamma \vdash M : A$ and $\Gamma \rhd E : A$. The typing system contains an assumption rule, an introduction rule, an elimination rule and a rule for typing substitution. These are standard, except for the use of two sorts of sequents. The coercion rule changes the kind of sequent. The displayed formula of the coercion rule is the *coercion formula*. The construction $ap(E)$ ($=\{E/x\}x$) represents the inverse of the coercion rule.

Reduction rules: The reduction rules of λ_{Nat} will act on the head of substitutions $\{hd(M)N_1...N_m/x\}P$. In order to have access to such heads, it is convenient to introduce the following syntactic expressions:

$$\mathcal{C} ::= \{[]/x\}P \,|\, N \cdot \mathcal{C}$$

These expressions are called *meta-contexts* of λ_{Nat}. As opposed to the contexts of λ^{Gtz}, which are formal expressions of λ^{Gtz}, meta-contexts are not formal expressions of λ_{Nat}, but rather a device in the meta-language. Intuitively, a meta-context is a substitution with a "hole": $\{[]N_1...N_k/x\}P$. Formally, given E, we define $\mathcal{C}[E]$ (the result of filling E in the hole of $\mathcal{C}$) by recursion on $\mathcal{C}$: $(\{[]/x\}P)[E] = \{E/x\}P$ and $(N \cdot \mathcal{C})[E] = \mathcal{C}[EN]$. So informally $N \cdot \mathcal{C}$ can be thought of as the meta-context $\mathcal{C}[[]N]$.

In addition to $\lambda x.\_$ and $(x)\_$, there is in meta-contexts a new binder (over x) $\{[]/x\}\_$. A variable x is a free variable of $\mathcal{C}$ (notation $x \in \mathcal{C}$) if x occurs in $\mathcal{C}$, but not in the scope of a binder over x.

The reduction rules of λ_{Nat} are as follows:

$$\begin{array}{llll} (\beta) & \mathcal{C}[hd(\lambda x.M)N] \to \langle N/x\rangle(\mathcal{C}[hd(M)]) & (\sigma) & \langle M/x\rangle P \to [M/x]P \\ (\pi) & \mathcal{C}[hd(\{E/x\}P)] \to \{E/x\}(\mathcal{C}[hd(P)]) & (\mu) & \{E/x\}(\mathcal{C}[hd(x)]) \to \mathcal{C}[E],\ x \notin \mathcal{C} \end{array}$$

The first three reduction rules, β, π and σ, enforcing every head to be of the form $hd(x)$ and to be in the function position of some application (hence not to be the actual-parameter position of some substitution). The $\beta\pi\sigma$-normal forms are given by:

$$\begin{array}{rcl} M, N, P & ::= & x \,|\, \lambda x.M \,|\, \{EN/x\}P \\ E & ::= & hd(x) \,|\, EN \end{array}$$

Later on, we will refer to this set as $\boxed{\text{A}}$.

By *normalisation* we mean $\beta\pi\sigma$-reduction. At the level of derivations, the *normality criterion* is: a derivation in λ_{Nat} is $\beta\pi\sigma$-normal if every coercion formula occurring in it is an assumption and the main premiss of an elimination. This extends von Plato's criterion of normality. Indeed, if m is always 1 in $\{hd(M)N_1...N_m/x\}P$, coercion formula = main premiss of elimination, and the criterion boils down to: the main premiss of an elimination is an assumption.

The particular case $P = x$ of rule σ reads $ap(hd(M)) \to M$ and is named ϵ. There is a fourth reduction rule, named μ, which is a handy tool not available in λg. Consider the λ-term xNN', that is, $ap(hd(ap(hd(x)N))N')$. After a π step we get $\{hd(x)N/z\}\{hd(z)N'/z'\}z'$ (z, z' fresh), which is a $\beta\pi\sigma$-normal form, if N, N' are. After a μ step one gets $ap(hd(x)NN')$, which is much simpler.

Related systems: A term M is $\beta\pi\sigma$-normal and only contains substitutions of the form $ap(E)$ iff M is a normal term of Cervesato and Pfenning's coercion calculus in [1]. Later on, we will refer to the class of such terms as $\boxed{\text{B}}$. They are given by

$$\begin{array}{rcl} M,N & ::= & x \mid \lambda x.M \mid ap(EN) \\ E & ::= & hd(x) \mid EN \ . \end{array}$$

Another set, essentially equivalent to this one, is the set of β-normal forms of $\lambda\mathcal{N}$, a coercion calculus studied in [7].

Fragments of λ_{Nat} are determined by placing restrictions on the number m in $\{hd(M)N_1...N_m/x\}P$ - recall Fig. 2. There is a $m \geq 1$-fragment, in fact system λNm of [11]. Another, important fragment is described next.

The $m \leq 1$-fragment: The fragment is closed under

$$\begin{array}{lll} (\beta) & \mathcal{C}[hd(\lambda x.M)N] \to \langle N/x\rangle(\mathcal{C}[hd(M)]), & \text{where } \mathcal{C} \text{ has the form } \{[]/y\}P \\ (\sigma) & \langle M/x\rangle P \to [M/x]P & \\ (\pi) & \mathcal{C}[hd(\{E/x\}P)] \to \{E/x\}(\mathcal{C}[hd(P)]), & \text{where } \mathcal{C} \text{ has the form } N' \cdot \{[]/y\}P' \\ & & \text{and } E \text{ is of the form } hd(M)N. \end{array}$$

In π we are avoiding the cases $\mathcal{C} = \{[]/y\}P'$ and $E = hd(M)$, which would not violate the $m \leq 1$ restriction. By doing this, we are, for simplicity, just presenting the call-by-name version of the fragment, where critical pairs between π and σ do not arise. Also some special cases of μ-reduction exist inside the fragment, but they happen to be simultaneously σ-reductions. So we do not add a μ-rule.

This fragment gives a version λ_{gs} of the λg-calculus with explicit substitution λgs. The β-rule of λgs is recovered as follows. Let $\mathcal{C} = \{[]/y\}P$. Then

$$\begin{array}{rcl} (\lambda x.M)(N, y.P) & = & \{hd(\lambda x.M)N/y\}P \\ & = & \mathcal{C}[hd(\lambda x.M)N] \\ & \to_\beta & \langle N/x\rangle\mathcal{C}[hd(M)] \\ & = & \langle N/x\rangle\langle M/y\rangle P \ . \end{array}$$

As to the π-rule of λgs, let $E = hd(M)N$ and $\mathcal{C} = N' \cdot \{[]/y\}P'$. Then

$$\begin{array}{rcl} M(N, x.P)(N', y.P') & = & \{hd(\{hd(M)N/x\}P)N'/y\}P' \\ & = & \mathcal{C}[hd(\{E/x\}P)] \\ & \to_\pi & \{E/x\}\mathcal{C}[hd(P)] \\ & = & \{hd(M)N/x\}\{hd(P)N'/y\}P' \\ & = & M(N, x.P(N', y.P')) \ . \end{array}$$

The $m \leq 1$-terms are the terms normal w.r.t. the following *permutation rule*

$$(\nu) \qquad \{ENN'/y\}P \to \{EN/z\}\{hd(z)N'/y\}P \ ,$$

with $z \notin N', P$. Notice that $\nu \subseteq \mu^{-1}$. Clearly, ν is terminating (the number of ν-redexes decreases at each step). Also local confluence is easily checked. The ν-nf of M is written $\nu(M)$.

Combining restrictions $m \geq 1$ and $m \leq 1$, we arrive at the $m = 1$-fragment, where substitutions have the form $\{hd(M)N/x\}P$, that is, the form of generalised application. It should be obvious that the $m = 1$-fragment is nothing but a version λ_g of λg.

4 Isomorphism

Mappings Ψ and Θ: We start with a mapping $\Psi : \lambda_{\mathsf{Nat}} - Terms \longrightarrow \lambda^{\mathsf{Gtz}} - Terms$. Let $\Psi(M) = t$, $\Psi(N_i) = u_i$ and $\Psi(P) = v$. The idea is to map, say, $\{hd(M)N_1N_2N_3/x\}P$ to $t(u_1 :: u_2 :: u_3 :: (x)v)$. This is achieved with the help of an auxiliary function $\Psi : \lambda_{\mathsf{Nat}} - EEs \times \lambda^{\mathsf{Gtz}} - Contexts \longrightarrow \lambda^{\mathsf{Gtz}} - Terms$ as follows:

$$\begin{array}{rclrcl} \Psi(x) &=& x & \Psi(hd(M), k) &=& (\Psi M)k \\ \Psi(\lambda x.M) &=& \lambda x.\Psi M & \Psi(EN, k) &=& \Psi(E, \Psi N :: k) \\ \Psi(\{E/x\}P) &=& \Psi(E, (x)\Psi P) & & & \end{array}$$

Next we consider a mapping $\Theta : \lambda^{\mathsf{Gtz}} - Terms \longrightarrow \lambda_{\mathsf{Nat}} - Terms$. Let $\Theta(t) = M$, $\Theta(u_i) = N_i$ and $\Theta(v) = P$. The idea is to map, say, $t(u_1 :: u_2 :: u_3 :: (x)v)$ to $\{hd(M)N_1N_2N_3/x\}P$. This is achieved with the help of an auxiliary function $\Theta : \lambda_{\mathsf{Nat}} - EEs \times \lambda^{\mathsf{Gtz}} - Contexts \longrightarrow \lambda_{\mathsf{Nat}} - Terms$ as follows:

$$\begin{array}{rclrcl} \Theta(x) &=& x & \Theta(E, (x)v) &=& \{E/x\}\Theta v \\ \Theta(\lambda x.t) &=& \lambda x.\Theta t & \Theta(E, u :: k) &=& \Theta(E\Theta u, k) \\ \Theta(tk) &=& \Theta(hd(\Theta t), k) & & & \end{array}$$

Contexts vs meta-contexts: Let $MetaContexts$ be the set of meta-contexts of λ_{Nat}. It is obvious that there is a connection between contexts of λ^{Gtz} and meta-contexts of λ_{Nat}. There is a function $\Theta_{\_} : Contexts \rightarrow MetaContexts$ defined by $\Theta_{(x)v} = \{[]/x\}\Theta v$ and $\Theta_{u::k} = \Theta u \cdot \Theta_k$, and a function $\Psi_{\_} : MetaContexts \rightarrow Contexts$ defined by $\Psi_{\{[]/x\}P} = (x)\Psi P$ and $\Psi_{N \cdot \mathcal{C}} = \Psi N :: \Psi_{\mathcal{C}}$.

We can identify each meta-context $\mathcal{C}$ of λ_{Nat} with a function of type $EEs \rightarrow Substs$, where $Substs$ is the set $\{M \in \lambda_{\mathsf{Nat}} : M$ is of the form $\{E/x\}P\}$; it is the function that sends E to $\mathcal{C}[E]$ (hence $\mathcal{C}(E) = \mathcal{C}[E]$). Now let k be a context of λ^{Gtz} and consider $\Theta(\_, k) : EEs \rightarrow Substs$. By induction on k one proves easily that $\Theta(\_, k)$ and Θ_k are the same function, *i.e.*

$$\Theta_k[E] = \Theta(E, k) \quad . \tag{2}$$

Theorem 2 (Isomorphism). *Mappings Ψ and Θ are sound, mutually inverse bijections between the set of λ^{Gtz}-terms and the set of λ_{Nat}-terms. Moreover, for each $R \in \{\beta, \sigma, \pi, \mu\}$:*

1. *$t \rightarrow_R t'$ in λ^{Gtz} iff $\Theta t \rightarrow_R \Theta t'$ in λ_{Nat}.*

2. $M \to_R M'$ *in* λ_{Nat} *iff* $\Psi M \to_R \Psi M'$ *in* λ^{Gtz}.

Proof: For bijection, prove $\Theta\Psi M = M$ and $\Theta\Psi(E,k) = \Theta(E,k)$ by simultaneous induction on M and E, and prove $\Psi\Theta t = t$ and $\Psi\Theta(E,k) = \Psi(E,k)$, by simultaneous induction on t and k. It follows that $k = \Psi_{\mathcal{C}}$ iff $\mathcal{C} = \Theta_k$. As to isomorphism, the "if" statements follow from the "only if" statements and bijection.

The "only if" statement 1 is proved together with the claim that, if $k \to_R k'$ in λ^{Gtz}, then, for all E, $\Theta_k[E] \to_R \Theta_{k'}[E]$ in λ_{Nat}. The proof is by simultaneous induction on $t \to_R t'$ and $k \to_R k'$, and uses the following properties of Θ: (i) if $\Theta(E',k) = \Theta(E,(x)v)$ then $\Theta(E', k@k') = \{E/x\}\Theta(hd(\Theta v), k')$; (ii) $\Theta(\langle u/x\rangle t) = \langle \Theta u/x\rangle\Theta t$; (iii) $\Theta([u/x]t) = [\Theta u/x]\Theta t$. Here are the base cases:

Case β.

$$\begin{tikzcd}
(\lambda x.t)(u :: k) \arrow[r, mapsto] \arrow[d, "\beta"'] & \Theta((\lambda x.t)(u :: k)) \arrow[r, equal] & \Theta_k[hd(\lambda x.\Theta t)\Theta u] \arrow[d, "\beta"] \\
\langle u/x\rangle(tk) \arrow[r, mapsto] & \Theta(\langle u/x\rangle(tk)) \arrow[r, equal] & \langle \Theta u/x\rangle\Theta_k[hd(\Theta t)]
\end{tikzcd}$$

$$\begin{aligned}
\Theta((\lambda x.t)(u :: k)) &= \Theta(\lambda x.\Theta t, u :: k) && \text{(by def. of } \Theta)\\
&= \Theta_{u::k}[hd(\lambda x.\Theta t)] && \text{(by (2))}\\
&= (\Theta u \cdot \Theta_k)[hd(\lambda x.\Theta t)] && \text{(by def. of } \Theta_{\_\_})\\
&= \Theta_k[hd(\lambda x.\Theta t)\Theta u] && \text{(by def of } \mathcal{C}[E])
\end{aligned}$$

$$\begin{aligned}
\Theta(\langle u/x\rangle(tk)) &= \langle \Theta u/x\rangle\Theta(tk) && \text{(by (ii))}\\
&= \langle \Theta u/x\rangle\Theta(hd(\Theta t), k) && \text{(by def. of } \Theta)\\
&= \langle \Theta u/x\rangle\Theta_k[hd(\Theta t)] && \text{(by (2))}
\end{aligned}$$

Case π. Suppose $\Theta(hd(\Theta t), k) = \Theta(E, (x)v)$.

$$\begin{tikzcd}
(tk)k' \arrow[r, mapsto] \arrow[d, "\pi"'] & \Theta((tk)k') \arrow[r, equal] & \Theta_{k'}[hd(\{E/x\}\Theta v)] \arrow[d, "\pi"] \\
t(k@k') \arrow[r, mapsto] & \Theta(t(k@k')) \arrow[r, equal] & \{E/x\}\Theta_{k'}[hd(\Theta v)]
\end{tikzcd}$$

$$\begin{aligned}
\Theta((tk)k') &= \Theta(hd(\Theta(hd(\Theta t), k)), k') && \text{(by def. of } \Theta)\\
&= \Theta_{k'}[hd(\Theta(hd(\Theta t), k))] && \text{(by (2))}\\
&= \Theta_{k'}[hd(\Theta(E, (x)v))] && \text{(by assumption)}\\
&= \Theta_{k'}[hd(\{E/x\}\Theta v)] && \text{(by def. of } \Theta)
\end{aligned}$$

$$\begin{aligned}
\Theta(t(k@k')) &= \Theta(hd(\Theta t), k@k') && \text{(by def. of } \Theta)\\
&= \{E/x\}\Theta(hd(\Theta v), k') && \text{(by (i))}\\
&= \{E/x\}\Theta_{k'}[hd(\Theta v)] && \text{(by (2))}
\end{aligned}$$

Case σ: $\Theta(\langle t/x\rangle v) \overset{(ii)}{=} \langle \Theta t/x\rangle \Theta v \to_\sigma [\Theta t/x]\Theta v \overset{(iii)}{=} \Theta([t/x]v)$.
Case μ:

$$\begin{tikzcd}
(x)xk \arrow[r, mapsto] \arrow[d, "\mu"'] & \Theta_{(x)xk}[E] \arrow[r, equal] & \{E/x\}\Theta_k[hd(x)] \arrow[d, "\mu"] \\
k \arrow[rr, mapsto] & & \Theta_k[E]
\end{tikzcd}$$

$$\begin{aligned}
\Theta_{(x)xk}[E] &= \{E/x\}\Theta(xk) && \text{(by def. } \Theta_{\_}\text{)} \\
&= \{E/x\}\Theta(hd(x),k) && \text{(by def. of } \Theta\text{)} \\
&= \{E/x\}\Theta_k[hd(x)] && \text{(by (2))}
\end{aligned}$$

The "only if" statement 2 is proved together with the claim that, if $E \to_R E'$ in λ_{Nat}, then, for all k, $\Psi(E,k) \to_R \Psi(E',k)$ in λ^{Gtz}. The proof is by simultaneous induction on $t \to_R t'$ and $E \to_R E'$, and uses the following properties of Ψ: (i) if $\Psi(E,k'') = \Psi(hd(M),k')$ then $\Psi(E,k''@k) = \Psi(M)(k'@k)$; (ii) $\Psi(\langle N/x\rangle M) = \langle \Psi N/x\rangle \Psi M$; (iii) $\Psi([N/x]M) = [\Psi N/x]\Psi M$. We also need the following remark:

$$\mathcal{C} = \Theta_k \Rightarrow \Psi(\mathcal{C}[E]) = \Psi(E,k)\ . \tag{3}$$

Indeed, if $\mathcal{C} = \Theta_k$, then, by (2), $\mathcal{C} = \Theta(\_,k)$. Hence, $\Psi(\mathcal{C}[E]) = \Psi(\Theta(E,k)) = \Psi(E,k)$. Here are the base cases of the inductive proof:

Case β: Let $\mathcal{C} = \Theta_k$.

$$\begin{tikzcd}
\mathcal{C}[hd(\lambda x.M)N] \arrow[r, mapsto] \arrow[d, "\beta"'] & \Psi(\mathcal{C}[hd(\lambda x.M)N]) \arrow[r, equal] & (\lambda x.\Psi M)(\Psi N :: k) \arrow[d, "\beta"] \\
\langle N/x\rangle \mathcal{C}[hd(M)] \arrow[r, mapsto] & \Psi(\langle N/x\rangle \mathcal{C}[hd(M)]) \arrow[r, equal] & \langle \Psi N/x\rangle(\Psi(M)k)
\end{tikzcd}$$

$$\begin{aligned}
\Psi(\mathcal{C}[hd(\lambda x.M)N]) &= \Psi(hd(\lambda x.M)N,k) && \text{(by (3))} \\
&= (\lambda x.\Psi M)(\Psi N :: k) && \text{(by def. of } \Psi\text{)}
\end{aligned}$$

$$\begin{aligned}
\Psi(\langle N/x\rangle \mathcal{C}[hd(M)]) &= \langle \Psi N/x\rangle \Psi(\mathcal{C}[hd(M)]) && \text{(by (ii))} \\
&= \langle \Psi N/x\rangle \Psi(hd(M),k) && \text{(by (3))} \\
&= \langle \Psi N/x\rangle(\Psi(M)k) && \text{(by def. of } \Psi\text{)}
\end{aligned}$$

Case π: Let $\mathcal{C} = \Theta_k$. Suppose $\Psi(E,(x)\Psi P) = \Psi(hd(M),k')$.

$$\begin{tikzcd}
\mathcal{C}[hd(\{E/x\}P)] \arrow[r, mapsto] \arrow[d, "\pi"'] & \Psi(\mathcal{C}[hd(\{E/x\}P)]) \arrow[r, equal] & (\Psi(M)k')k \arrow[d, "\pi"] \\
\{E/x\}\mathcal{C}[hd(P)] \arrow[r, mapsto] & \Psi(\{E/x\}\mathcal{C}[hd(P)]) \arrow[r, equal] & (\Psi M)(k'@k)
\end{tikzcd}$$

$$\begin{aligned}
\Psi(\mathcal{C}[hd(\{E/x\}P)]) &= \Psi(hd(\{E/x\}P), k) && \text{(by (3))}\\
&= \Psi(E, (x)\Psi P)k && \text{(by def. of } \Psi)\\
&= \Psi(hd(M), k')k && \text{(by assumption)}\\
&= (\Psi(M)k')k && \text{(by def. of } \Psi)
\end{aligned}$$

$$\begin{aligned}
\Psi(\{E/x\}\mathcal{C}[hd(P)]) &= \Psi(E, (x)\Psi(\mathcal{C}[hd(P)])) && \text{(by def. of } \Psi)\\
&= \Psi(E, (x)\Psi(hd(P), k)) && \text{(by (3))}\\
&= \Psi(E, (x)\Psi(P)k) && \text{(by def. of } \Psi)\\
&= \Psi(E, ((x)\Psi(P))@k) && \text{(by def. of @)}\\
&= (\Psi M)(k'@k) && \text{(by (i))}
\end{aligned}$$

Case σ: $\Psi(\langle M/x\rangle P) \overset{(ii)}{=} \langle \Psi M/x\rangle \Psi P \to_\sigma [\Psi M/x]\Psi P \overset{(iii)}{=} \Psi([M/x]P)$.
Case μ: Let $\mathcal{C} = \Theta_k$.

$$\begin{tikzcd}
\{E/x\}\mathcal{C}[hd(x)] \arrow[r, mapsto] \arrow[d, "\mu"'] & \Psi(\{E/x\}\mathcal{C}[hd(x)]) \arrow[r, equal] & \Psi(E, (x)xk) \arrow[d, "\mu"] \\
\mathcal{C}[E] \arrow[rr, mapsto] & & \Psi(E, k)
\end{tikzcd}$$

$$\begin{aligned}
\Psi(\{E/x\}\mathcal{C}[hd(x)]) &= \Psi(E, (x)\Psi(\mathcal{C}[hd(x)])) && \text{(by def. of } \Psi)\\
&= \Psi(E, (x)\Psi(hd(x), k)) && \text{(by (3))}\\
&= \Psi(E, (x)xk) && \text{(by def. of } \Psi)
\end{aligned}$$

■

Corollary 1 (SN). *Every typable $t \in \lambda_{\mathsf{Nat}}$ is $\beta\pi\sigma\mu$-SN.*

Proof: From Theorems 1 and 2. ■

5 Analyzing the isomorphism

Inversion of associativity: By "applicative term" we mean the following data: a function (or head), m arguments ($m \geq 0$) and a continuation (or tail). The notion of applicative term is intended as a common abstraction to the notions of cut in λ^{Gtz}

$$t(u_1 :: ... :: u_m :: (x)v) \ , \tag{4}$$

and substitution in λ_{Nat}

$$\{hd(M)N_1...N_m/x\}P \ . \tag{5}$$

By allowing $m = 0$ we include in the notion of applicative term the extreme case when there is really no application, because there is no argument.

The presentation of sequent calculus and natural deduction as systems λ^{Gtz} and λ_{Nat}, respectively, reduces the difference between the two kinds of systems to the difference between two ways of organizing applicative terms. When (4) and (5) are regarded in the abstract way of just providing the data that constitutes an applicative term, the only difference that remains between the two expressions is that (4) associates to the right, so that the head t is at the surface and the continuation $(x)v$ is hidden at the bottom of the expression, whereas (5) associates to the left, so that the head $hd(M)$ is hidden at the bottom of the expression, and the continuation x, P is at the surface. The isomorphism $\lambda^{\mathsf{Gtz}} \cong \lambda_{\mathsf{Nat}}$ may, then, be described as a mere inversion of the *associativity* of applicative terms.

Interpretations of λ^{Gtz}: From the previous paragraph follows that an interpretation of λ^{Gtz} is as a λ-calculus with right associative applicative terms. Another interpretation is as a formalized meta-calculus for λ_{Nat} (and not for a smaller natural deduction system, like λg or λgs, let alone λ). Contexts in λ^{Gtz} are the formal counterpart to meta-contexts in λ_{Nat} and the interpretation of cut given by $\Theta(tk) = \Theta_k[hd(\Theta t)]$ corresponds to "fill Θt in the hole of Θ_k".

Variants of the isomorphism: $\lambda^{\mathsf{Gtz}} \cong \lambda_{\mathsf{Nat}}$ is a particular manifestation of the isomorphism between sequent calculus and natural deduction. For instance, if rule π of λ^{Gtz} is taken in the call-by-name version $(tk)(u :: k') \to t(k@(u :: k'))$ [2], avoiding a critical pair with σ, then there is corresponding version for rule π of λ_{Nat}, namely $\mathcal{C}[hd(\{E/x\}P)N] \to \{E/x\}(\mathcal{C}[hd(P)N])$.

Another variant of rule π is the "eager" variant, determined by a slight change in the definition of @: $((x)V)@k = (x)Vk$, if V is a value (*i.e.* variable or abstraction); and $((x)tk')@k = (x)t(k'@k)$. So, one keeps pushing k until a value is found.

Let $\{Es/xs\}P$ denote a sequence of substitutions $\{E_1/x_1\}...\{E_n/x_n\}P$. The eager variant of π for natural deduction is $\mathcal{C}[hd(\{Es/xs\}V)] \to \{Es/xs\}\mathcal{C}[hd(V)]$. So, the eager variant takes a sequence of substitutions out, as opposed to the lazy variant, which takes them one by one.

Theorem 2 still holds with eager π. In the proof of the "only if" part of statement 1, property (i) of Θ becomes slightly different: if $\Theta(E', k) = \{Es/xs\}V$ then $\Theta(E', k@k') = \{Es/xs\}\Theta(hd(V), k')$. As to the the "only if" part of statement 2, property (i) of Ψ holds again with eager definition of $k@k'$, but some more work is needed. Fix $\mathcal{C} = \Theta_k$. One proves that, if $\Psi(\{Es/xs\}V) = (\Psi M)k'$, then $\Psi(\{Es/xs\}\mathcal{C}[V]) = (\Psi M)(k'@k)$. The proof has two cases, $n = 1$ and $n > 1$, where n is the number of substitutions in $\{Es/xs\}V$. The first case is consequence of property (i). The second uses an auxiliary lemma: $((z)\Psi(\{Es/xs\}V))@k = (z)\Psi(\{Es/xs\}\mathcal{C}[V])$. This auxiliary lemma, in turn, follows from: $((z)\Psi(E, k'))@k = (z)\Psi(E, k'@k)$.

Neutral fragments: The $m \leq 1$-fragment λ^{gs} of λ^{Gtz} and the $m \leq 1$-fragment λ_{gs} of λ_{Nat} are two copies of λgs, hence isomorphic. The isomorphism

$\lambda^{gs} \cong \lambda_{gs}$ is a degenerate form of Theorem 2, with Θ and Ψ translating between $t(x)v$ and $\{hd(M)/x\}P$, and between $t(u :: (x)v)$ and $\{hd(M)N/x\}P$.

$$\begin{tikzcd}
\lambda^{gs} \arrow[rr, shift left, "\Theta"] & & \lambda_{gs} \arrow[ll, shift left, "\Psi"] \\
& \lambda gs \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

We can see $t(x)v$ and $\{hd(M)/x\}P$ as two ways of *decomposing explicit substitution*:

$$\dfrac{\Gamma \vdash t : A \qquad \dfrac{\Gamma, x : A \vdash v : B}{\Gamma; A \vdash (x)v : B}\,\mathit{Selection}}{\Gamma \vdash t(x)v : B}\,\mathit{Cut}$$

$$\dfrac{\dfrac{\Gamma \vdash M : A}{\Gamma \rhd hd(M) : A}\,\mathit{Coercion} \qquad \Gamma, x : A \vdash P : B}{\Gamma \vdash \{hd(M)/x\}P}\,\mathit{Subst}$$

This sheds some light on the logical status of explicit substitution. Traditionally, explicit substitution has a hybrid proof-theoretical character, because it is regarded as a form of cut, even when considered in a natural deduction setting. The explanation of explicit substitution as a particular case of the more general constructor of primitive substitution $\{E/x\}P$ shows how it fits in a pure natural deduction setting.

In addition, $t(u :: (x)v)$ and $\{hd(M)N/x\}P$ are two ways of *decomposing generalised elimination*:

$$\dfrac{\Gamma \vdash t : A \supset B \qquad \dfrac{\Gamma \vdash u : A \qquad \dfrac{\Gamma, x : B \vdash v : C}{\Gamma; B \vdash (x)v : C}\,\mathit{Selection}}{\Gamma; A \supset B \vdash u :: (x)v : C}\,\mathit{Left}}{\Gamma \vdash t(u :: (x)v) : C}\,\mathit{Cut}$$

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash M : A \supset B}{\Gamma \rhd hd(M) : A \supset B}\,\mathit{Coercion} \qquad \Gamma \vdash N : A}{\Gamma \rhd hd(M)N : B}\,\mathit{Elim} \qquad \Gamma, x : B \vdash P : C}{\Gamma \vdash \{hd(M)N/x\}P : C}\,\mathit{Subst}$$

The latter decomposition is pleasing for two reasons. First, all constructor of λ_{Nat} different from x and $\lambda x.M$ (coercion, elimination, substitution) enter the decomposition of generalised application. Second, while $MN := M(N, x.x)$ says that general elimination is more general than ordinary elimination (ordinary elimination alone cannot express general elimination), the decomposition

$M(N, x.P) := \{hd(M)N/x\}P$ says that, in a sense, ordinary elimination is more primitive than generalised elimination.[5]

Examining the isomorphism $\lambda^{gs} \cong \lambda_{gs}$, we realise that the fragments λ^{gs} and λ_{gs} are mere notational variants of each other, to ways of writing the same combinatorial, abstract object. This combinatorial object lives in the intersection of λ^{Gtz} and λ_{Nat}, and we may say that it is *neutral* w.r.t the classification as a sequent calculus or natural deduction system. At first this seems paradoxical, because it entails that the λ-calculus is also neutral ($\lambda \subset \lambda g \subset \lambda gs$). But a second thought shows that the $m \leq 1$-fragments have to be neutral, because, when one is restricted to $m \leq 1$ arguments in applicative terms, one cannot tell whether the system has a left-associative or a right-associative structure.[6]

Fig. 3. Particular cases of the isomorphism and important classes of terms

$$\begin{tikzcd}
\lambda^{\mathsf{Gtz}} \arrow[rr, leftrightarrow, "\Psi{,}\Theta"] \arrow[dr, "\nu"] \arrow[dd, "\beta\pi\sigma"'] & & \lambda_{\mathsf{Nat}} \arrow[dl, "\nu"'] \arrow[dd, "\beta\pi\sigma"] \\
& \lambda gs \arrow[dd, "\beta\pi\sigma"] & \\
\boxed{\text{Schwichtenberg nfs}} \arrow[dr, "\nu"] \arrow[dd, dotted] & {} \arrow[l, "\Psi"'] \arrow[r, "\Theta"] & \boxed{\text{A}} \arrow[dl, "\nu"'] \arrow[dd, dotted] \\
& \boxed{\text{von Plato nfs}} \arrow[dd, dotted] & \\
\boxed{\text{Herbelin nfs}} \arrow[dr, "\nu"] & {} \arrow[l, "\Psi"'] \arrow[r, "\Theta"] & \boxed{\text{B}} \arrow[dl, "\nu"'] \\
& \boxed{\text{Mints nfs}} &
\end{tikzcd}$$

Particular cases of the isomorphism: We now analyze the diagram in Figure 3. The $m \leq 1$-fragment λ^{gs} of λ^{Gtz} and the $m \leq 1$-fragment λ_{gs} of λ_{Nat} are identified. In both cases, the fragment consists of the ν-nfs.The λ-calculus is absent from Figure 3 (λ-terms form a subset of λgs), but there are three sets in bijective correspondence with the set of β-normal λ-terms, namely $\boxed{\text{Herbelin nfs}}$,

[5] There are also inclusions of λg into the systems λGm and λNm of [11], based on the decompositions of generalised application as a right-principal cut + left-introduction or multiary generalised application + coercion.

[6] Similar taxonomical considerations have been made for the first time in [7], regarding smaller fragments of sequent calculus and natural deduction.

$\boxed{\text{B}}$ and $\boxed{\text{Mints nfs}}$, *i.e.* the lower triangle. $\boxed{\text{Herbelin nfs}} \cong \boxed{\text{Mints nfs}}$ was known [5], the bijection being the restriction of ν to $\boxed{\text{Herbelin nfs}}$. A simple form of Theorem 2 is $\boxed{\text{Herbelin nfs}} \cong \boxed{\text{B}}$. The latter bijection extends to another bijection, namely $\boxed{\text{Schwichtenberg nfs}} \cong \boxed{\text{A}}$, whereas $\boxed{\text{Herbelin nfs}} \cong \boxed{\text{Mints nfs}}$ doesn't extend, because many "multiary" cut-free derivations in $\boxed{\text{Schwichtenberg nfs}}$ have the same ν-normal form in $\boxed{\text{von Plato nfs}}$. The bijection $\boxed{\text{Schwichtenberg nfs}} \cong \boxed{\text{A}}$ is in turn the residue of the isomorphism $\lambda^{\mathsf{Gtz}} \cong \lambda_{\mathsf{Nat}}$, because it is the bijection between the sets of $\beta\pi\sigma$-nfs. The dotted arrows represent three reduction relations generated by permutative conversions. Two of such relations have been characterised [5, 26].

6 Unification

A problem of wrong focus: Recall that applicative terms in λ^{Gtz} and λ_{Nat} have the form of cuts and substitutions

$$t(u_1 :: ... :: u_m \cdot (x)v) \ , \tag{6}$$
$$\{hd(M)N_1...N_m/x\}P \ . \tag{7}$$

respectively. In both cases there is a head, m arguments ($m \geq 0$) 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 context k; in the second case, the term is split just before the tail, with the rest of data organized as an elimination expression E. In the first case, the head is focused, in the second it is the tail that is focused.

Now both cut-elimination and 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 β and π are non-local reduction rules in the natural deduction system λ_{Nat}.

Non-local rules are bad, for instance, for the implementation of λ_{Nat}, where the search for heads has to be made explicitly. The solution we propose is to extend λ_{Nat} to a calculus where applicative terms are split at arbitrary position, and not just around the tail. This means that both elimination expressions E and contexts k are used in the representation of applicative terms, and this representation turns out to unify both cuts and substitutions.

The telescopic effect: The idea of manipulating elimination expressions E and contexts k in the same system has another motivation. Let M_0 be the substitution (7), and let $\Theta t = M$, $\Theta u_i = N_i$ and $\Theta v = P$. There are m choices of E, k such that $M_0 = \Theta(E, k)$, ranging from the choice $E = hd(M)N_1...N_m$ and $k = (x)v$ to the choice $E = hd(M)$ and $k = u_1 :: ... :: u_m :: (x)v$. This last case is particularly important, because the representation of application M_0 as $\Theta(hd(M), k)$ brings to the surface the head $hd(M)$. In general, we will use pattern matching of substitution with $\Theta(hd(M), k)$ to obtain the effect of extracting the head of the substitution, an effect we call the *telescopic* effect. Similarly one extracts the tail of cuts by pattern-matching with $\Psi(E,(x)v)$.

The telescopic effect is useful in making global rules look local. This is achieved by manipulating simultaneously, in the meta-language, both elimination expressions E and contexts k. For instance, reduction rule π in λ_{Nat} may be defined as follows:

$$(\pi) \qquad \Theta(hd(\{E/x\}P),k) \to \{E/x\}\Theta(hd(P),k) \ .$$

The calculus we introduce next manipulates expressions $\Theta(E,k)$ formally.

The unified calculus: Expressions in $\lambda\mathsf{U}$ are given by:

$$\begin{array}{lr} \text{(Terms)} & M,N,P ::= x \mid \lambda x.M \mid \underline{\theta}(E,K) \\ \text{(Elimination Expressions)} & E ::= hd(M) \mid EN \\ \text{(Contexts)} & L,K ::= (x)P \mid N :: K \end{array}$$

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

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

In $\lambda\mathsf{U}$, a *sequent term* is a term with no occurrences of EN, *i.e* an elimination-free term, whereas a *natural deduction term* is a term with no occurrences of $N :: K$, *i.e.* a left-introduction-free term. In sequent terms and natural deduction terms, unified cuts have the form

$$MK = \underline{\theta}(hd(M),K) \qquad\qquad \{E/x\}P = \underline{\theta}(E,(x)P) \ ,$$

respectively. These equations show how unified cut unifies cut and and substitution. Explicit substitution

$$M(x)P = \underline{\theta}(hd(M),(x)P) = \{hd(M)/x\}P \ ,$$

denoted $\langle M/x\rangle P$, is the intersection of cut and substitution. Sequent terms (resp. natural deduction terms) dispense with the syntactic class of elimination expressions E (resp. contexts K) and constitute a copy of λ^{Gtz}-terms (resp. λ_{Nat}-terms) in $\lambda\mathsf{U}$. Given a λ^{Gtz}-term t (resp. λ_{Nat}-term M), we denote by $t^{\diamond}$ (resp. $M^{\diamond}$) its copy in $\lambda\mathsf{U}$.

The reduction rules of $\lambda\mathsf{U}$ are as follows:

$$\begin{array}{lr} (\beta) & \underline{\theta}(hd(\lambda x.M), N :: K) \to \langle N/x\rangle\underline{\theta}(hd(M),K) \\ (\pi) & \underline{\theta}(hd(\underline{\theta}(E,L)),K) \to \underline{\theta}(E, L@K) \\ (\sigma) & \underline{\theta}(hd(N),(x)P) \to [N/x]P \\ (\underline{\psi}) & \underline{\theta}(EN,K) \to \underline{\theta}(E, N :: K) \end{array}$$

Meta-substitution $[N/x]P$ is defined in the expected way. So is append of contexts: for instance, $((x)P)@K = (x)\underline{\theta}(hd(P),K) = (x)(PK)$. Rules β, π, and σ require a head in focus; for this reason, they are local transformations. Rule $\underline{\psi}$ is a step towards focusing a head. A $\lambda\mathsf{U}$-term is a $\underline{\psi}$-nf iff it is a sequent term.

$\underline{\psi}$-reduction is terminating, because it decreases the number of occurrences of $\underline{EN}$. It is also locally confluent (to see this, let us call E,N and K the *components* of a $\underline{\psi}$-redex $\underline{\theta}(EN, K)$; if two distinct $\underline{\psi}$-redexes overlap, then one is a subexpression of one of the components of the other; thus, the contractions of the two redexes commute). So $\underline{\psi}$-reduction is confluent. We denote by $\underline{\psi}(M)$ the unique $\underline{\psi}$-nf of a λU-term M. $\underline{\psi}(M)$ is given by

$$\underline{\psi}(x) = x$$
$$\underline{\psi}(\lambda x.M) = \lambda x.\underline{\psi}M$$
$$\underline{\psi}(\underline{\theta}(E, K)) = \underline{\psi}(E, \underline{\psi}K)$$

$$\underline{\psi}(hd(M), K) = (\underline{\psi}M)K \qquad \underline{\psi}((x)P) = (x)\underline{\psi}P$$
$$\underline{\psi}(EN, K) = \underline{\psi}(E, \underline{\psi}N :: K) \qquad \underline{\psi}(N :: K) = \underline{\psi}N :: \underline{\psi}K$$

Lemma 1. *For all $M \in \lambda_{\mathsf{Nat}}$, $\underline{\psi}(M^\diamond) = \Psi(M)^\diamond$.*

Proof: Consider the claim: for all $E \in \lambda_{\mathsf{Nat}}$, $k \in \lambda^{\mathsf{Gtz}}$, $\Psi(E, k)^\diamond = \underline{\psi}(E^\diamond, k^\diamond)$. The two claims are proved by simultaneous induction on M and E. ■

It is easy to see that sequent terms are closed for $\beta\pi\sigma$-reduction, and a λ^{Gtz}-term t $\beta\pi\sigma$-reduces in λ^{Gtz} exactly as $t^\diamond$ $\beta\pi\sigma$-reduces in λU. Let us see what happens when one $\beta\pi\sigma$-reduces $M^\diamond$ in λU, for $M \in \lambda_{\mathsf{Nat}}$.

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

1. *In λU, if $M_0 \to_R M_1$ and $M_0 \to_{\underline{\psi}} M_2$ then there is M_3 such that $M_2 \to_R M_3$ and $M_1 \to^*_{\underline{\psi}} M_3$.*
2. *In λU, if $M \to_R N$, then $\underline{\psi}(M) \to_R \underline{\psi}(N)$.*
3. *If $M \to_R N$ in λ_{Nat}, then there are M_1, N_1 such that, in λU: $M_1 \to_R N_1$ and $M^\diamond \to^*_{\underline{\psi}} M_1$ and $N^\diamond \to^*_{\underline{\psi}} N_1$.*

Proof: First we consider statement 1. We prove simultaneously three statements of the form: if $\xi_0 \to_R \xi_1$ and $\xi_0 \to_{\underline{\psi}} \xi_2$ then there is ξ_3 such that $\xi_2 \to_R \xi_3$ and $\xi_1 \to^*_{\underline{\psi}} \xi_3$. One for terms $\xi_i = M_i$, another for elimination expressions $\xi_i = E_i$, and finally one for contexts $\xi_i = K_i$. The proof is by simultaneous induction on $M_0 \to_R M_1$, $E_0 \to_R E_1$, and $K_0 \to_R K_1$. We just illustrate one base case and one inductive case, namely those cases which have more subcases.

Case π. Suppose $M_0 = \underline{\theta}(hd(\underline{\theta}(E, L)), K) \to_\pi \underline{\theta}(E, L@K) = M_1$ and $M_0 \to_{\underline{\psi}} M_2$. There are three subcases. (i) The $\underline{\psi}$-reduction happens in E. Then, the two reduction steps commute. (ii) The $\underline{\psi}$-reduction happens in L or K. Then the two reduction steps commute, because, for all L, L', and K, $L \to_{\underline{\psi}} L'$ implies $L@K \to_{\underline{\psi}} L'@K$ and $K@L \to_{\underline{\psi}} K@L'$. (iii) $E = E_0N_0$ and $M_2 = \underline{\theta}(hd(\underline{\theta}(E_0, N_0 :: L)), K)$. Then take $M_3 = \underline{\theta}(E_0, N_0 :: (L@K))$.

Case $M_0 = \underline{\theta}(E_0, K) \to_R \underline{\theta}(E_1, K) = M_1$, with $E_0 \to_R E_1$. Suppose $M_0 \to_{\underline{\psi}} M_2$. There are three subcases. (i) The $\underline{\psi}$-reduction happens in the K. Then the two reduction steps commute. (ii) The $\underline{\psi}$-reduction happens in the E_0. Then apply the induction hypothesis. (iii) $E_0 = EN$ and $M_2 = \underline{\theta}(E, N :: K)$. From

$E_0 \to_R E_1$ and $E_0 = EN$ we get either $E \to_R E'$ and $E_1 = E'N$ (in this case take $M_3 = \underline{\theta}(E', N :: K)$) or $N \to_R N'$ and $E_1 = EN'$ (in this case take $M_3 = \underline{\theta}(E, N' :: K)$). This concludes the proof of statement 1.

Statement 2 is a corollary of statement 1 and the fact that $P \to^*_{\underline{\psi}} \underline{\psi}(P)$.

Finally we consider statement 3. First we define, for each meta-context $\mathcal{C}$ of λ_{Nat}, a context $\mathcal{C}^\diamond$ of $\lambda\mathsf{U}$ by recursion on $\mathcal{C}$: $(\{[]/x\}P)^\diamond = (x)P^\diamond$ and $(N \cdot \mathcal{C})^\diamond = N^\diamond :: \mathcal{C}^\diamond$. Next

$$(\mathcal{C}[E])^\diamond \to^*_{\underline{\psi}} \underline{\theta}(E^\diamond, \mathcal{C}^\diamond) \qquad (*)$$

is proved by induction on $\mathcal{C}$. Statement 3 is proved simultaneously with another statement: if $E \to_R F$ in λ_{Nat}, then there are E_1, F_1 such that, in $\lambda\mathsf{U}$: $E_1 \to_R F_1$ and $E^\diamond \to^*_{\underline{\psi}} E_1$ and $F^\diamond \to^*_{\underline{\psi}} F_1$. The proof is by simultaneous induction on $M \to_R N$ and $E \to_R F$. The inductive cases are straightforward. We just illustrate one of the base cases.

Case β. Suppose $M = \mathcal{C}[hd(\lambda x.P)Q] \to_\beta \langle Q/x\rangle\mathcal{C}[hd(P)] = N$. Then

$$\begin{array}{rll}
M^\diamond & = \mathcal{C}[hd(\lambda x.P)Q]^\diamond & \\
& \to^*_{\underline{\psi}} \underline{\theta}((hd(\lambda x.P)Q)^\diamond, \mathcal{C}^\diamond) & \text{(by (*))} \\
& = \underline{\theta}(hd(\lambda x.P^\diamond)Q^\diamond, \mathcal{C}^\diamond) & \text{(by def. of } (\_)^\diamond) \\
& \to_{\underline{\psi}} \underline{\theta}(hd(\lambda x.P^\diamond), Q^\diamond :: \mathcal{C}^\diamond) & \\
& \to_\beta \langle Q^\diamond/x\rangle\underline{\theta}(hd(P^\diamond), \mathcal{C}^\diamond) & \\
& = \langle Q^\diamond/x\rangle\underline{\theta}(hd(P)^\diamond, \mathcal{C}^\diamond) & \text{(by def. of } (\_)^\diamond) \\
& \leftarrow^*_{\underline{\psi}} \langle Q^\diamond/x\rangle\mathcal{C}[hd(P)]^\diamond & \text{(by (*))} \\
& \leftarrow^*_{\underline{\psi}} (\langle Q/x\rangle\mathcal{C}[hd(P)])^\diamond & \text{(by def. of } (\_)^\diamond) \\
& = N^\diamond &
\end{array}$$

So one may take $M_1 = \underline{\theta}(hd(\lambda x.P^\diamond), Q^\diamond :: \mathcal{C}^\diamond)$ and $N_1 = \langle Q^\diamond/x\rangle\underline{\theta}(hd(P^\diamond), \mathcal{C}^\diamond)$. ■

Theorem 3 (Normalisation in $\lambda\mathsf{U}$). *Suppose $M_1 \to_{R_1} M_2 \to (\cdots) \to M_n \to_{R_n} M_{n+1}$ is a $\beta\pi\sigma$-reduction sequence in λ_{Nat}. Then, the reductions in $\lambda\mathsf{U}$ depicted in Fig. 4 hold, when vertical arrows denote $\underline{\psi}$-reduction.*

Proof: By induction on n. Case $n = 1$. Suppose $M_1 \to_{R_1} M_2$. The diagram

$$\begin{tikzcd}
M_1^\diamond \arrow[d, dotted] \arrow[r, phantom, "\boxed{1}"] & M_2^\diamond \arrow[d, dotted] \\
\bullet \arrow[r, "R_1"'] \arrow[d, dotted] \arrow[r, phantom, "\boxed{2}", shift right=4ex] & \bullet \arrow[d, dotted] \\
\underline{\psi}(M_1^\diamond) \arrow[r, "R_1"'] & \underline{\psi}(M_2^\diamond)
\end{tikzcd}$$

holds, as $\boxed{1}$ (resp. $\boxed{2}$) follows from part 3 (resp. part 2) of Proposition 2.

Fig. 4. Normalisation in $\lambda\mathsf{U}$

$$\begin{tikzcd}
M_1 \arrow[r, "R_1"] & M_2 \arrow[r, "R_2"] & M_3 \arrow[r] & (\cdots) \arrow[r] & M_n \arrow[r, "R_n"] & M_{n+1}
\end{tikzcd}$$

$$\begin{tikzcd}
M_1^\diamond \arrow[d, dotted, two heads] & M_2^\diamond \arrow[d, dotted, two heads] & M_3^\diamond \arrow[dd, dotted, two heads] & (\cdots) & M_n^\diamond \arrow[ddd, dotted, two heads] & M_{n+1}^\diamond \arrow[ddd, dotted, two heads] \\
\bullet \arrow[r, "R_1"'] \arrow[dddd, dotted, two heads] & \bullet \arrow[d, dotted, two heads] & & & & \\
 & \bullet \arrow[r, "R_2"'] \arrow[ddd, dotted, two heads] & \bullet \arrow[ddd, dotted, two heads] & (\cdots) & & \\
 & & & & \bullet \arrow[r, "R_n"'] \arrow[d, dotted, two heads] & \bullet \arrow[d, dotted, two heads] \\
\underline{\psi}(M_1^\diamond) \arrow[r, "R_1"] & \underline{\psi}(M_2^\diamond) \arrow[r, "R_2"] & \underline{\psi}(M_3^\diamond) \arrow[r] & (\cdots) \arrow[r] & \underline{\psi}(M_n^\diamond) \arrow[r, "R_n"] & \underline{\psi}(M_{n+1}^\diamond)
\end{tikzcd}$$

Inductive case. Suppose $M_1 \to_{R_1} (\cdots) \to M_n \to_{R_n} M_{n+1} \to_{R_{n+1}} M_{n+2}$. The relevant part of the induction hypothesis gives:

$$\begin{tikzcd}
M_n^\diamond \arrow[d, dotted, two heads] & M_{n+1}^\diamond \arrow[d, dotted, two heads] \\
\bullet \arrow[r, "R_n"'] \arrow[d, dotted, two heads] & N \arrow[d, dotted, two heads] \\
\underline{\psi}(M_n^\diamond) \arrow[r, "R_1"'] & \underline{\psi}(M_{n+1}^\diamond)
\end{tikzcd}$$

We want

$$\begin{tikzcd}
M_n^\diamond \arrow[d, dotted] & M_{n+1}^\diamond \arrow[d, dotted] & M_{n+2}^\diamond \arrow[dd, dotted] \\
\bullet \arrow[r, "R_n"'] \arrow[dd, dotted] & N \arrow[d, dotted] & \\
 & P \arrow[r, "R_{n+1}"'] \arrow[d, dotted] & Q \arrow[d, dotted] \\
\underline{\psi}(M_n^\diamond) \arrow[r, "R_n"'] & \underline{\psi}(M_{n+1}^\diamond) \arrow[r, "R_{n+1}"'] & \underline{\psi}(M_{n+2}^\diamond)
\end{tikzcd}$$

This diagram follows from the following

$$\begin{tikzcd}
M_n^\diamond \arrow[dd, dotted] & M_{n+1}^\diamond \arrow[dd, dotted] \arrow[dr, dotted] & & M_{n+2}^\diamond \arrow[dr, dotted] & \boxed{1} \\
 & \boxed{2} & \bullet \arrow[rr, "R_{n+1}"'] \arrow[dd, dotted] & & \bullet \arrow[dd, dotted] \\
\bullet \arrow[r, "R_n"'] \arrow[ddd, dotted] & N \arrow[ddd, dotted] \arrow[dr, dotted] & & \boxed{4} & \\
 & \boxed{3} & P \arrow[rr, "R_{n+1}"] \arrow[ddl, dotted] & & Q \arrow[ddl, dotted] \\
 & & & \boxed{5} & \\
\underline{\psi}(M_n^\diamond) \arrow[r, "R_n"'] & \underline{\psi}(M_{n+1}^\diamond) \arrow[rr, "R_{n+1}"'] & & \underline{\psi}(M_{n+2}^\diamond) &
\end{tikzcd}$$

First we build $\boxed{1}$, using part 3 of Proposition 2. Next we use confluence of $\underline{\psi}$-reduction twice, to obtain successively $\boxed{2}$ and $\boxed{3}$. Finally, we apply successively parts 1 and 2 of Proposition 2 to obtain $\boxed{4}$ and $\boxed{5}$. ■

Regarding Fig. 4 again, we can now compare reduction of M_1 in λ_{Nat} with reduction of $M_1^\diamond$ in $\lambda\mathsf{U}$. The latter is obtained from the former by interleaving $\underline{\psi}$-reduction steps. To a possibly non-local reduction step $\to_{R_i}$ in the former corresponds a necessarily local reduction step $\to_{R_i}$ in the latter. The interleaved $\underline{\psi}$-reduction steps do explicitly the focusing of heads implicit in the reduction steps at the λ_{Nat} level. The reduction of $\underline{\psi}(M_1^\diamond)$ is morally the same as the reduction of $\Psi(M_1)$ in λ^{Gtz}. Fig. 4 is a refinement of the "only if" part of statement 1 in Theorem 2.

Finally, observe that part 2 of Proposition 2 allows the projection of $\beta\pi\sigma\underline{\psi}$-reduction sequences of $\lambda\mathsf{U}$ into $\beta\pi\sigma$-reduction sequences of λ^{Gtz}. So, the lifting of Theorem 1 from λ^{Gtz} to $\lambda\mathsf{U}$ is immediate, having in mind, additionally, that $\underline{\psi}$-reduction is terminating.

Theorem 4. *$\beta\pi\underline{\psi}$-reduction in $\lambda\mathsf{U}$ is s.n. on typable terms.*

Variant of the unified calculus: Consider the permutative conversions p_i of λJ^m [13] (which in turn are a variant of the permutative conversions of [26]), given here for λ^{Gtz} with the help of telescopic effect:

$$\begin{array}{lrcl}
(p_1) & \Psi(EN,(x)y) & \to & y,\ \text{if } x \neq y \\
(p_2) & \Psi(EN,(x)\lambda y.t) & \to & \lambda y.\Psi(EN,(x)t) \\
(p_3) & \Psi(EN,(x)\Psi(E',(y)v)) & \to & \Psi(p_3(E,N,x,E'),(y)v),\ \text{if } x \notin v,
\end{array}$$

where $p_3(E,N,x,E')$ is defined by recursion on E' as follows: $p_3(E,N,x,hd(M)) = hd(\Theta(EN,(x)\Psi M))$ and $p_3(E,N,x,E'N') = p_3(E,N,x,E')\Theta(EN,(x)\Psi N')$.

For instance, if $x \notin v$, then

$$t_1(u_1 :: (x)t_2(u_2 :: (y)v)) \to_{p_3} t_1(u_1 :: (x)t_2)(t_1(u_1 :: (x)u_2) :: (y)v)\ .$$

Indeed, let $E = hd(\Theta t_1)$, $N = \Theta u_1$, and $F = hd(\Theta t_2)\Theta u_2$. Then,

$$\begin{array}{lll}
 & t_1(u_1 :: (x)t_2(u_2 :: (y)v)) & \\
= & \Psi(EN,(x)\Psi(F,(y)v)) & \text{(telescopic effect)} \\
\to_{p_3} & \Psi(p_3(E,N,x,F),(y)v) & \\
= & \Psi(hd(\Theta(EN,(x)t_2))\Theta(EN,(x)u_2),(y)v) & \text{(def. of } p_3) \\
= & \Psi(EN,(x)t_2)(\Psi(EN,(x)u_2) :: (y)v) & \text{(def. of } \Psi \text{ and } \Psi\Theta(\_,\_) = \Psi(\_,\_)) \\
= & t_1(u_1 :: (x)t_2)(t_1(u_1 :: (x)u_2) :: (y)v) & \text{(telescopic effect)}
\end{array}$$

The rules p_i act at the tail of applicative terms, and aim at reducing terms $\Psi(E,(x)v)$ to the form $\Psi(E,(x)x)$. They are non-local rules in λ^{Gtz}, contrary to the corresponding rules in λ_{Nat}. In λ_{Nat} no telescopic effect is needed, one simply defines $\{EN/x\}\lambda y.M \to \lambda y.\{EN/x\}M$, etc.

A variant of the unified calculus can be defined as an extension of λ^{Gtz} where means are available for defining in a local way tail-active conversions.

$$\begin{array}{rl}
\text{(Terms)} & t,u,v ::= x \mid \lambda x.t \mid \underline{\psi}(e,k) \\
\text{(Elimination Expressions)} & e,f ::= hd(t) \mid eu \\
\text{(Contexts)} & k ::= (x)v \mid u :: k
\end{array}$$

We use lower case meta-variables to emphasize that this calculus is to be regarded as an extension of λ^{Gtz}. $\underline{\psi}(e,k)$ is now a formal expression of the calculus, the unified cut whose focus we now regard to be k. Of course, something new only happens when we replace reduction rule $\underline{\psi}$ by a new rule $\underline{\theta}$ for bringing continuations to focus:

$$(\underline{\theta}) \qquad \underline{\psi}(e,u :: k) \to \underline{\psi}(eu,k)$$

The p rules now read:

$$\begin{array}{lrcl}
(p_1) & \underline{\psi}(eu,(x)y) & \to & y,\ \text{if } x \neq y \\
(p_2) & \underline{\psi}(eu,(x)\lambda y.t) & \to & \lambda y.\underline{\psi}(eu,(x)t) \\
(p_3) & \underline{\psi}(eu,(x)\underline{\psi}(f,(y)v)) & \to & \underline{\psi}(p_3(e,u,x,f),(y)v),\ \text{if } x \notin v,
\end{array}$$

where $p_3(e,u,x,f)$ is defined by recursion on f as follows: $p_3(e,u,x,hd(t)) = hd(\underline{\psi}(eu,(x)t))$ and $p_3(e,u,x,fv) = p_3(e,u,x,f)\underline{\psi}(eu,(x)v)$.

This time, a natural deduction term p-reduces in the unified calculus exactly as it p-reduces in λ_{Nat}, and the p-reduction of sequent terms in the unified calculus is interleaved with $\underline{\theta}$-steps, and has the ladder shape of Fig. 4. For instance, let $e = hd(t_1)$, $u = u_1$, and $f = hd(t_2)u_2$. Then,

$$\begin{array}{lll}
 & t_1(u_1 :: (x)t_2(u_2 :: (y)v)) & \\
= & \underline{\psi}(hd(t_1), u_1 :: (x)\underline{\psi}(hd(t_2), u_2 :: (y)v)) & \text{(abbreviation)} \\
\to^2_{\underline{\theta}} & \underline{\psi}(eu, (x)\underline{\psi}(f, (y)v)) & \\
\to_{p_3} & \underline{\psi}(p_3(e,u,x,f), (y)v) & \\
= & \underline{\psi}(hd(\underline{\psi}(eu,(x)t_2))\underline{\psi}(eu,(x)u_2), (y)v) & \text{(def. of } p_3\text{)} \\
\leftarrow_{\underline{\theta}} & \underline{\psi}(hd(\underline{\psi}(eu,(x)t_2)), \underline{\psi}(eu,(x)u_2) :: (y)v) & \\
\leftarrow^2_{\underline{\theta}} & \underline{\psi}(hd(\underline{\psi}(hd(t_1), u_1 :: (x)t_2)), \underline{\psi}(hd(t_1), u_1 :: (x)u_2) :: (y)v) & \\
= & t_1(u_1 :: (x)t_2)(t_1(u_1 :: (x)u_2) :: (y)v) & \text{(abbreviation)}
\end{array}$$

So, instead of telescopic effect (*i.e.* the search for tails in the meta-level), one has explicit $\underline{\theta}$-steps for focusing tails.

7 Related and future work:

Natural deduction: One of the contributions of this paper is a new system of natural deduction that combines the idea of coercion calculus with the idea of generalised elimination rule. In one sense, von Plato's work goes much farther than this paper, in that [27] covers the whole language of first order logic; on the other hand, it lacks an analysis of the correspondence between cut-elimination and normalisation. For the logic under analysis here, von Plato's work is supplemented. Not only we extended and refined system λg (and here it is quite appealing that we end up in a system where generalised application is decomposed into *modus ponens* and substitution), but also we give the precise connection between generalised normalisation and cut-elimination, which is this: system λgs, is the common core of cut-elimination in λ^{Gtz} and normalisation in λ_{Nat} - in particular, it is a fragment of the former.[7]

Once one has the natural deduction system λ_{Nat}, one can clarify the connection between cut and substitution, and translate between sequent calculus and natural deduction in a way that the classical mappings of Gentzen [15] and Prawitz [25] never could.

Cut vs substitution, left introduction vs elimination, cut-elimination vs normalisation: There is an entanglement in the traditional mappings between natural deduction and sequent calculus. An elimination is translated as a combination of cut and left introduction [15] and a left introduction is translated as a combination of elimination and meta-substitution [25]. With these mappings

[7] Similarly, system λg is the common core of cut-elimination in λGm and normalisation in λNm - in particular, it is a fragment of the former.

one proves that normalisation is a *"homomorphic"* image of cut-elimination [28, 24].[8]

The typing system of λ_{Nat} clarifies the puzzling relation between cut and substitution. Consider rule *Cut* in λ^{Gtz} and rule *Subst* in λ_{Nat}. First, we observe, as Negri and von Plato in [22], that the right cut-formula of *Cut*, but not the right substitution formula in *Subst*, may be the conclusion of a sequence of left introductions. Second, and here comes the novelty, we may also observe that the left substitution formula in *Subst*, but not left cut-formula in *Cut*, may be the conclusion of a sequence of elimination rules. So, cut is more general on the right, whereas substitution is more general on the left.

Mapping Θ (as well as it inverse Ψ) inverts the associativity of applicative terms, as envisaged by Herbelin, and establishes bijective correspondences between occurrences of left introduction $u :: k$ (resp. of cut tk) in the source term and occurrences of elimination EN (resp. of substitution $\{E/x\}P$) in the source term (inversely for Ψ). So the entanglement of traditional mappings is solved, and normalization (in λ_{Nat}) becomes the *isomorphic* image, under Θ, of cut-elimination (in λ^{Gtz}). This result improves, for the logic examined here, the classical results of Zucker and Pottinger [28, 24].

Herbelin's programme: The system λ_{Nat} also allows us to complete what we have called in [9] and in the introduction of this paper Herbelin's programme. As compared to [16], we covered full sequent calculus, where the constraints on left introduction that define Herbelin's fragment *LJT* are dropped. Already [1] and [7, 8] considered implicitly Herbelin's programme. The improvement over these works is that the spine calculus of [1], when restricted to the logic of this paper, and the sequent systems in [7, 8] are all fragments of Herbelin's *LJT* and, therefore, are under the restrictions already mentioned.

Unity of structural proof theory: Negri and von Plato dedicate a chapter of [22] to "diversity and unity in structural proof theory". As a way of restoring the unity of structural proof theory, they propose the "uniform" calculus, a system generalising both sequent calculus and the system of natural deduction with generalised elimination rules. $\lambda\mathsf{U}$ achieves a similar goal but with a radically different approach. While the approach in the uniform calculus is to retain generalised elimination rules and extend introduction rules to "generalised introduction rules", the approach in $\lambda\mathsf{U}$ is to break the generalised elimination rule and unify one of its components, substitution, with cut.

Applications and future work: An issue that deserves further consideration is the use of languages λ^{Gtz} and λ_{Nat} in practice. As emphasized in [1], the spine calculus, Herbelin's $\overline{\lambda}$ and - we add - λ^{Gtz}, give a useful representation of λ-terms for procedures that act on the head of applicative terms, like normalisation or unification. Conversely, λ_{Nat} gives a useful representation of λ-terms for procedures that act on the tail of applicative terms, like permutative conversion. It seems that the role of languages like λ^{Gtz} or λ_{Nat} is not as languages in which someone writes his programs, but either as internal languages for sym-

[8] For a study of the traditional mappings between sequent calculus and natural deduction, and some of their optimizations, see [10].

bolic systems, like theorem provers, or as intermediate languages for compilers of functional languages. On the other hand, languages λ^{Gtz} and λ_{Nat} are good tools for doing proof theory efficiently, as this paper shows.

We have seen that $\lambda\mathsf{U}$ is useful as a natural deduction system where the search for heads is made explicit. But if searching for heads is all we want, it is better to use λ^{Gtz} instead, where heads are always available. The real use for $\lambda\mathsf{U}$ is in hybrid situations, for instance, when one wants to study the interaction between cut-elimination and permutative conversions [12]. A simpler example is the μ rule, given next with telescopic effect:

$$\Theta(E,(x)\Theta(hd(x),k)) \to \Theta(E,k), \text{ if } x \notin k \ .$$

Observe 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 $\underline{\theta}(EN,K) = \underline{\theta}(E,N :: K)$.

8 Conclusions

Let us conclude with some reflection on two topics concerning the relationship between the λ-calculus and structural proof theory, specifically: (i) the λ-calculus (in an extended sense) as a vehicle for expressing the unity of structural proof theory; and conversely, (ii) the apparent diversity, and the deep unity of structural proof theory as determining the internal structure of the λ-calculus.

Radical answers: One way of looking at the Curry-Howard correspondence is as a methodology for giving systems of formal deduction a uniform presentation in the technical framework of typed λ-calculi. Such a presentation has a specific, fixed treatment of certain technical matters (what objects are derived? how are the structural rules dealt with?) which are thus abstracted away when two presentations are compared. This is why one finds in Herbelin's paper [16] - one of the first papers where a clear understanding of the Curry-Howard for sequent calculus is achieved - an immediate, sharp indication of what the difference between sequent calculus and natural deduction amounts to, only possible because the fog of irrelevant, little differences had been cleared away.

Herbelin's seminal suggestion in [16] is that the (computational) difference between sequent calculus and natural deduction may be reduced to a mere *question of representation* of λ-terms, when these are conceived in a sufficiently extended sense. This suggestion is fully supported and developed here, where we proposed an abstract, robust extension of the concept of λ-term (under two concrete representations: λ^{Gtz}-terms and λ_{Nat}-terms), and the concomitant typing and reduction machinery.

Such reduction of the difference between sequent calculus and natural deduction to a question of representation allows a radical answer to long-standing problems of structural proof-theory: if normalisation is extended as we propose in λ_{Nat}, then the meaning of $\lambda^{\mathsf{Gtz}} \cong \lambda_{\mathsf{Nat}}$ is that cut-elimination and normalisation are really the same process, they only look different because they operate with different representations of the same objects.

This answer shows how deep is the unity of structural proof theory, but immediately leads to other observations: the natural deduction representation, implemented in λ_{Nat}, is, from the point of view of executing normalisation, simply wrong, because it focuses on the wrong end of applicative terms (exactly as the sequent calculus representation, implemented in λ^{Gtz}, is wrong, from the point of view of implementing tail-active permutative conversions).

One finds syntactic remedies for the defects of the systems λ_{Nat} and λ^{Gtz}, if the two proof-system are unified into a single one, a process which, again, may be described as, and reduced to, a further extension of the concept of λ-term, as implemented in $\lambda\mathsf{U}$. So, remedies are found by realizing that the unity of structural proof theory is even deeper; and systems λ_{Nat} and λ^{Gtz}, together with their defects, are just the result of taking partial views of that unity.

Fig. 5. Unifying systems and neutral fragments

$$\begin{tikzcd}
 & \lambda\mathsf{U} & \\
\lambda^{\mathsf{Gtz}} \arrow[ur] & \lambda U \arrow[u] & \lambda_{\mathsf{Nat}} \arrow[ul] \\
\lambda Gm \arrow[u] \arrow[ur] & \lambda gs \arrow[ul] \arrow[ur] & \lambda Nm \arrow[ul] \arrow[u] \\
 & \lambda g \arrow[ul] \arrow[u] \arrow[ur] & \\
 & \lambda \arrow[u] &
\end{tikzcd}$$

Associativity dilemma: Fig. 5 collects the extensions of λ-calculus found in the present proof-theoretical investigation. Fig. 5 updates Fig. 1 by including the neutral fragments λg (which subsumes λ) and λgs (but notice that arrows for the isomorphism Θ, Ψ are omitted). Another interpretation of Fig. 5 is as showing different views of the concept of the λ-calculus, determined by various treatments of applications and substitution, among which the ordinary λ-calculus (with its ordinary treatment of application and substitution) is just a particular instance.

The question we address now is: how much of the architecture of Fig 5 is determined by the alternative between sequent calculus and natural deduction? Of course one might immediately say that such architecture holds also in the untyped setting, where the question is meaningless. So we reformulate the question by replacing the dilemma between sequent calculus and natural deduction by the more general dilemma between a right-associative and a left-associative implementation of the multiarity facility.

Disregarding λ, we have a cube where each system in the upper face (*e.g.* λgs) is a version with explicit substitution of the corresponding system in the lower face (*e.g.* λg). Starting at λgs and moving upwards, there are two alternative systems implementing the multiarity facility (λ^{Gtz} or λ_{Nat}), and then a system where these alternatives somehow coexist ($\lambda\mathsf{U}$). Similarly if we start at λg.

At the left column we have the right-associative systems (λ^{Gtz}, λGm), at the right column the left-associative systems (λ_{Nat}, λNm), and in the central column the systems that are largely insensitive to the associativity dilemma. As we said before, the neutral fragments λgs and λg have too restricted sets of terms, where the dilemma is not observable. In the unifying systems $\lambda\mathsf{U}$ and λU, the dilemma is partially avoided because the set of terms is sufficiently large to comprehend both a right-associative system and a left-associative system. However, for instance in $\lambda\mathsf{U}$, one still has a choice to make: one can give the left-to-right orientation to the equation

$$\underline{\theta}(EN, K) = \underline{\theta}(E, N :: K) \ , \tag{8}$$

which gives $\lambda\mathsf{U}$ the character of an extension of λ_{Nat}, or the opposite orientation, as in the variant of $\lambda\mathsf{U}$ proposed above, which gives the system the character of an extensions of λ^{Gtz}.

In some sense, equation (8) concentrates the whole associativity dilemma (in particular, the sequent calculus/natural deduction dilemma). There is an easy way of seeing this. We have argued that the fragments λ^{gs} and λ_{gs} should be identified (as we did in Fig. 5). This is the same to say that, in the definition of mappings Θ and Ψ, the only equations that do any work are $\Theta(E, u :: k) = \Theta(E\Theta u, k)$ and $\Psi(EN, k) = \Psi(E, \Psi N :: k)$; now in $\lambda\mathsf{U}$ these two equations collapse to (8). Only a variant of $\lambda\mathsf{U}$ where $\beta\pi\sigma$-reduction is defined modulo equation (8) - a system where there is simultaneous access to the head and the tail of applicative terms - would be truly insensitive to the associativity dilemma.

Acknowledgments: The author is supported by FCT, through Centro de Matemática, Universidade do Minho. We have used Paul Taylor's macros for typesetting diagrams.

References

1. I. Cervesato and F. Pfenning. A linear spine calculus. *Journal of Logic and Computation*, 13(5):639–688, 2003.

2. P.-L. Curien and H. Herbelin. The duality of computation. In *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming (ICFP '00), Montreal, Canada, September 18-21, 2000*, SIGPLAN Notices 35(9), pages 233–243. ACM, 2000.
3. H. B. Curry and R. Feys. *Combinatory Logic.* Noth Holland, Amsterdam, 1958.
4. R. Dyckhoff and L. Pinto. Cut-elimination and a permutation-free sequent calculus for intuitionistic logic. *Studia Logica*, 60:107–118, 1998.
5. R. Dyckhoff and L. Pinto. Permutability of proofs in intuitionistic sequent calculi. *Theoretical Computer Science*, 212:141–155, 1999.
6. J. Espírito Santo. Revisiting the correspondence between cut-elimination and normalisation. In *Proceedings of ICALP'2000*, volume 1853 of *Lecture Notes in Computer Science*, pages 600–611. Springer-Verlag, 2000.
7. 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/.
8. J. Espírito Santo. An isomorphism between a fragment of sequent calculus and an extension of natural deduction. In M. Baaz and A. Voronkov, editors, *Proceedings of LPAR'02*, volume 2514 of *Lecture Notes in Artificial Intelligence*, pages 354–366. Springer-Verlag, 2002.
9. J. Espírito Santo. Completing Herbelin's programme. In S. Ronchi Della Rocca, editor, *Proceedings of TLCA'07*, volume 4583 of *Lecture Notes in Computer Science*, pages 118–132. Springer-Verlag, 2007.
10. J. Espírito Santo. Delayed substitutions. In Franz Baader, editor, *Proceedings of RTA'07*, Lecture Notes in Computer Science, pages 169–183. Springer-Verlag, 2007.
11. J. Espírito Santo. Refocusing generalised normalisation. In S.B. Cooper, B. Lowe, and A. Sorbi, editors, *Proceedings of CiE'07*, volume 4497 of *Lecture Notes in Computer Science*, pages 258–267. Springer-Verlag, 2007.
12. J. Espírito Santo, M. J. Frade, and Luís Pinto. Structural proof theory as rewriting. In F. Pfenning, editor, *Proc. of RTA'06*, volume 4098 of *Lecture Notes in Computer Science*, pages 197–211. Springer-Verlag, 2006.
13. 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.
14. J. Gallier. *Logic for Computer Science: Foundations of Automated Theorem Proving.* Wiley, 1986.
15. G. Gentzen. Investigations into logical deduction. In M. E. Szabo, editor, *The collected papers of Gerhard Gentzen*, pages 68–131. North Holland, 1969.
16. 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.
17. W. A. Howard. The formulae-as-types notion of construction. In J. P. Seldin and J. R. Hindley, editor, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 480–490. Academic Press, New York, 1980.
18. 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.
19. S. Kleene. *Introduction to metamathematics.* North Holland, 1952.
20. D. Miller, G. Nadathur, F Pfenning, and A. Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51:125–157, 1991.

21. G. Mints. Normal forms for sequent derivations. In P. Odifreddi, editor, *Kreiseliana*, pages 469–492. A. K. Peters, Wellesley, Massachusetts, 1996.
22. S. Negri and J. von Plato. *Structural Proof Theory*. Cambridge, 2001.
23. 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*, pages 190–201. Springer Verlag, 1992.
24. G. Pottinger. Normalization as a homomorphic image of cut-elimination. *Annals of Mathematical Logic*, 12:323–357, 1977.
25. D. Prawitz. *Natural Deduction. A Proof-Theoretical Study*. Almquist and Wiksell, Stockholm, 1965.
26. H. Schwichtenberg. Termination of permutative conversions in intuitionistic Gentzen calculi. *Theoretical Computer Science*, 212:247–260, 1999.
27. J. von Plato. Natural deduction with general elimination rules. *Annals of Mathematical Logic*, 40(7):541–567, 2001.
28. J. Zucker. The correspondence between cut-elimination and normalization. *Annals of Mathematical Logic*, 7:1–112, 1974.