Completing Herbelin's programme

José Espírito Santo

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

Abstract. In 1994 Herbelin started and partially achieved the programme of showing that, for intuitionistic implicational logic, there is a Curry-Howard interpretation of sequent calculus into a variant of the λ-calculus, specifically a variant which manipulates formally "applicative contexts" and inverts the associativity of "applicative terms". Herbelin worked with a fragment of sequent calculus with constraints on left introduction. In this paper we complete Herbelin's programme for full sequent calculus, that is, sequent calculus without the mentioned constraints, but where permutative conversions necessarily show up. This requires the introduction of a lambda-like calculus for full sequent calculus and an extension of natural deduction that gives meaning to "applicative contexts" and "applicative terms". Such extension is a calculus with *modus ponens* and primitive substitution that refines von Plato's natural deduction; it is also a "coercion calculus", in the sense of Cervesato and Pfenning. The proof-theoretical outcome is noteworthy: the puzzling relationship between cut and substitution is settled; and cut-elimination in sequent calculus is proven isomorphic to normalisation in the proposed natural deduction system. The isomorphism is the mapping that inverts the associativity of applicative terms.

1 Introduction

Herbelin's CSL'94 paper [11] is an integrated contribution into two closely related subjects: structural proof theory and the study of the computational interpretation of sequent calculus. Here, structural proof theory is taken in the restricted sense of the study of the relationship between natural deduction and sequent calculus, the two kinds of proof systems introduced since the subject was born [10]. Such relationship is a puzzle that constantly attracted attention during the last 70 years [10, 18, 22, 17, 14, 20]. The study of the computational interpretation of sequent calculus, with the purpose of extending the Curry-Howard correspondence, is a relatively recent topic, with the first explicit contributions starting in the early 1990's [9, 21, 13]. An integrated contribution to the two subjects is desirable: one should understand the differences and similarities between natural deduction and sequent calculus, if one wants to extend the Curry-Howard correspondence; and a way of expressing those differences and similarities is, precisely, via the corresponding computational interpretations.

Herbelin's paper initiates the programme of defining a λ-calculus (with a strongly normalising set of reduction rules) such that, by means of the calculus, the following two goals are achieved simultaneously: (1) to give a convincing computational interpretation of (a fragment of) sequent calculus, along the lines of the Curry-Howard correspondence; and (2) to express the difference between sequent calculus and natural deduction, reducing it to the mere inversion of the associativity of applicative terms.

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 contributions, 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 concludes that the difference between sequent calculus and natural deduction resides in the organization of applicative terms: sequent calculus is right-associative $t(u_1 :: ...(u_n :: []))$, whereas natural deduction is left-associative $(...(MN_1)...N_m)$.

Herbelin's paper achieved (1) for LJT and has the merit of suggesting that (2) can be achieved. Verification of (2) happened in later papers. The mapping that inverts the associativity of applicative terms is proved in [3] to be a bijection between normal λ-terms and cut-free $\overline{\lambda}$-terms, in [5] to be an isomorphism between the λ-calculus and a fragment of $\overline{\lambda}$, and in [6] to be an isomorphism between an extension of the λ-calculus and a larger fragment of $\overline{\lambda}$. Fulfillment of (2) is useful for (1), because only an isomorphic natural deduction system gives rigorous meaning to "applicative context" and "applicative term".

Notwithstanding the parts of Herbelin's programme already completed (including the extension of (1) to classical logic in [2]), a lot remains unfinished. 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. In addition, the fulfilment of (2), in connection with larger fragments of sequent calculus, requires the extension of the natural deduction system. One idea for this extension is in [6], and turns out to be the idea of defining natural deduction as a "coercion calculus", in the sense of Cervesato and Pfenning [1]. Another idea is that of generalised elimination rules, due to von Plato [20].

In the setting of intuitionistic implicational logic, we contribute to the completion of 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 is in terms of a λ-calculus λ^{Gtz} with a primitive notion of applicative context, taken in a natural, generalised sense. In order to fulfil (2), a system of natural deduction λ_{Nat} is defined that extends and refines von Plato's natural deduction. It is a calculus with *modus ponens* and primitive substitution and it is also a coercion calculus. Then we prove that $\lambda^{\mathsf{Gtz}} \cong \lambda_{\mathsf{Nat}}$ 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" $\lambda\mathtt{s}$ of [7]; strong normalisation for λ_{Nat} follows by isomorphism. These results constitute, for the logic under analysis here, considerable improvements over [11, 1, 20, 6].

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 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 introduced here are denoted λ_S; more or less traditional systems of 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.

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

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*. 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: $[] = (x)x$, $[u_1, ..., u_n] = u_1 :: ...u_n :: []$, and $\langle u/x\rangle t = u(x)t$.

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

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

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

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

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

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. An implicit contraction may happen here.

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

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

where

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

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:

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

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 [16], 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 $\lambda\mathtt{s}$-calculus of [7].

The terms of $\lambda\mathtt{s}$ 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 $(\_)^* : \lambda^{\mathsf{Gtz}} \to \lambda\mathtt{s}$. More precisely, mappings $(\_)^* : \lambda^{\mathsf{Gtz}} - Terms \to \lambda\mathtt{s} - Terms$ and $(\_,\_)^* : \lambda\mathtt{s} - Terms \times \lambda^{\mathsf{Gtz}} - Contexts \to \lambda\mathtt{s} - Terms$ 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 $\lambda\mathtt{s}$.*

Proof: Follows from the following four facts: (i) $(\langle N/x\rangle M, k)^* \to^+_\pi \langle N/x\rangle(M,k)^*$; (ii) $((M,k)^*, k')^* \to^+_\pi (M, append(k,k'))^*$; (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: [7] proves that every typable $t \in \lambda\mathtt{s}$ is $\beta\pi\sigma$-SN (if we use for $\lambda\mathtt{s}$ 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:

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

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 $(y)v$. These correspond also to von Plato's "fully normal " natural deductions. $\beta\pi\sigma$-normal forms correspond exactly to Schwichtenberg's *multiary* cut-free terms [19]. We refer to these as *Schwichtenberg nfs.*

A context $u_1 :: ... :: u_m :: (x)x$ $(m \geq 0)$ is called an *applicative* context, and may be regarded as a list $[u_1, ..., u_m]$, if we think of $(x)x$ as $[]$. If every context in a term t is applicative, t is a $\overline{\lambda}$-term. A term t is $\beta\pi\sigma$-normal and only contains

applicative contexts iff t is a cut-free $\overline{\lambda}$-term, in the sense of [11]. We refer to such terms as *Herbelin nfs*. They are given by $t, u ::= x \mid \lambda x.t \mid x(u :: k)$ and $k ::= [] \mid u :: k$. Another characterisation of this set is as the set of Schwichtenberg's terms (1) normal w.r.t. certain permutative conversions [19].

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. There is a $m \geq 1$-fragment, which gives a version of the system λJ^m studied in [8]. There is a $m \leq 1$-fragment, which gives a version λ^{gs} of the λg-calculus with explicit substitution λgs, to be defined in the next section. The $m \leq 1$-terms are the terms normal w.r.t. the following *permutation rule*

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

with $z \notin v, k$. Notice that $\to_\nu \subseteq \to_\mu^{-1}$. Clearly, ν is terminating and locally confluent. The ν-nf of t is written $\nu(t)$.

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 [20] may be presented as a type system for the λ-calculus with generalized application. The latter is the system ΛJ of [12], 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 $\beta\pi$-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. A $\beta\pi$-normal form M is called a *Mints normal form* if, for every application $x(N, y.P)$ in M, P is y-normal [4]. 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 of Mints nfs is as $\beta\pi$-normal forms which are, in addition, normal w.r.t. a set of permutation rules given in [4].

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 λgs-term is in $\beta\pi\sigma$-normal form iff it is a λg-term in $\beta\pi$-normal form.[2]

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 set of β-normal λ-terms is in bijective correspondence with the set of Mints normal forms [14, 4].

Motivation for λ_{Nat}: If one sees generalised application $M(N, x.P)$ as a substitution $subst(MN, x.P)$ (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 $subst(subst(MN, x.x)N', y.y)$, with x, y fresh. A solution is to allow $m \geq 0$ application as actual parameters of substitutions: $subst(MN_1...N_m, x.P)$. The particular case $m = 0$ encompasses explicit substitution. The usefulness of allowing $m > 1$ is precisely in having the alternative way $subst(MNN', x.x)$ of writing MNN'.

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

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

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* $hd(M)$ (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 $\{hd(M)N_1...N_m/x\}P$, with $m \geq 0$. Generalised elimination is recovered as $\{hd(M)N/x\}P$, that is the particular case $m = 1$. Ordinary elimination is $\{hd(M)N/x\}x$. We will use the following abbreviations: $ap(E) = \{E/z\}z$, $MN = ap(hd(M)N)$ and $\langle N/x\rangle M = \{hd(N)/x\}M$.

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

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

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

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

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

[2] For a slightly different definition of λgs see [7].

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:

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

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 $N \cdot \mathcal{C}$ can be thought of as the meta-context $\mathcal{C}[[]N]$.

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

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

There are 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 in the actual-parameter position of some substitution). The $\beta\pi\sigma$-normal forms are given by:

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

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 xN_1N_2, that is, $ap(hd(ap(hd(x)N_1))N_2)$. After a π step we get $\{hd(x)N_1/z_1\}\{hd(z_1)N_2/z_2\}z_2$ (z_i's fresh), which is a $\beta\pi\sigma$-normal form, if N_1, N_2 are. After a μ step one gets $ap(hd(x)N_1N_2)$, 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 $M, N ::= x \mid \lambda x.M \mid ap(EN)$ and $E ::= hd(x) \mid EN$. Another characterisation of this set is as the set of β-normal forms of $\lambda\mathcal{N}$, a coercion calculus studied in [5].

Fragments of λ_{Nat} are determined by placing restrictions on the number m in $\{hd(M)N_1...N_m/x\}P$. There is a $m \leq 1$-fragment, which 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 $\{hd(\lambda x.M)N/y\}P = \mathcal{C}[hd(\lambda x.M)N] \rightarrow_\beta \langle N/x\rangle\mathcal{C}[hd(M)] = \langle N/x\rangle\langle M/y\rangle P$. The π-rule of λgs is recovered as follows. Let $E = hd(M)N$ and $\mathcal{C} = N' \cdot \{[]/y\}P'$. Then $\{hd(\{hd(M)N/x\}P)N'/y\}P' = \mathcal{C}[hd(\{E/x\}P)] \rightarrow_\pi \{E/x\}\mathcal{C}[hd(P)] = \{hd(M)N/x\}\{hd(P)N'/y\}P'$.

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

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

with $z \notin N', P$. Notice that $\nu \subseteq \mu^{-1}$. Clearly, ν is terminating and locally confluent. The ν-nf of M is written $\nu(M)$.

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:

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

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:

$$\begin{array}{rlrl} \Theta(x) &= x & \Theta'(E, (x)v) &= \{E/x\}\Theta v \\ \Theta(\lambda x.t) &= \lambda x.\Theta t & \Theta'(E, u :: k) &= \Theta'(E\Theta u, k) \\ \Theta(tk) &= \Theta'(hd(\Theta t), k) & & \end{array}$$

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 $\{M \in \lambda_{\mathsf{Nat}} : M \text{ is of the form } \{E/x\}P\}$; 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.* $\Theta_k[E] = \Theta'(E, k)$.

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. *$t \to_R t'$ in λ^{Gtz} iff $\Theta t \to_R \Theta t'$ 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. We just sketch the "only if" statement 1, which 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.

$$\Theta((\lambda x.t)(u :: k)) = \Theta_{u::k}[hd(\lambda x.\Theta t)] = (\Theta u \cdot \Theta_k)[hd(\lambda x.\Theta t)] = \Theta_k[hd(\lambda x.\Theta t)\Theta u]$$
$$\downarrow$$
$$\Theta(\langle u/x\rangle(tk)) \overset{(ii)}{=} \langle \Theta u/x\rangle \Theta(tk) = \langle \Theta u/x\rangle \Theta_k[hd(\Theta t)]$$

Case π. Suppose $\Theta'(hd(\Theta t),k) = \Theta'(E,(x)v)$.

$$\Theta((tk)k') = \Theta_{k'}[hd(\Theta'(hd(\Theta t),k))] = \Theta_{k'}[hd(\Theta'(E,(x)v))] = \Theta_{k'}[hd(\{E/x\}\Theta v)]$$
$$\downarrow$$
$$\Theta(t(k@k')) = \Theta'(hd(\Theta t), k@k') \overset{(i)}{=} \{E/x\}\Theta'(hd(\Theta v),k') = \{E/x\}\Theta_{k'}[hd(\Theta v)]$$

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: $\Theta_{(x)xk}[E] = \{E/x\}\Theta(xk) = \{E/x\}\Theta_k[hd(x)] \to \Theta_k[E]$. ■

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

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 [10] and a left introduction is translated

as a combination of elimination and meta-substitution [18]. With these mappings one proves that normalisation is a *"homomorphic"* image of cut-elimination [22, 17].[3]

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 [15], 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 Ψ establishes bijective correspondences between occurrences of elimination EN (resp. of substitution $\{E/x\}P$) in the source term and occurrences of left introduction $u :: k$ (resp. of cut tk) in the target term (inversely for Θ). So the entanglement of traditional mappings is solved, and the outcome is that normalization in λ_{Nat} becomes the *isomorphic* image, under Θ, of cut-elimination in λ^{Gtz}.

Applicative terms: 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". By "applicative term" we mean the following data: a function (or head), m arguments ($m \geq 1$) and a continuation (or tail). The notion of applicative term is intended as a common abstraction to the notions of cut in λ^{Gtz}

$$t(u_1 :: ... :: u_m :: (x)v) \ , \tag{2}$$

and substitution in λ_{Nat}

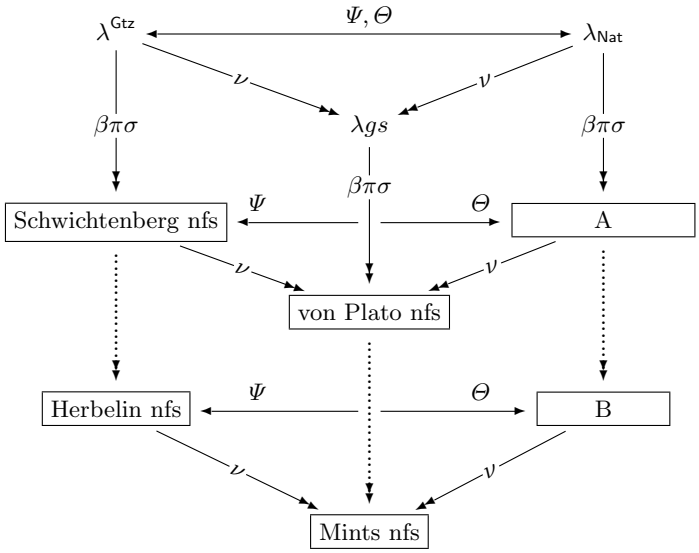
$$\{hd(M)N_1...N_m/x\}P \ . \tag{3}$$

When (2) and (3) 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 (2) 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 (3) 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 $\Theta(tk) = \Theta_k[hd(\Theta t)]$ is "fill Θt in the hole of Θ_k".

[3] For a study of the traditional mappings between sequent calculus and natural deduction, and some of their optimizations, see [7].

Fig. 1. Particular cases of the isomorphism and important classes of terms



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} $\mathcal{C}[hd(\{E/x\}P)N] \to \{E/x\}(\mathcal{C}[hd(P)N])$.

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 fact (i) becomes slightly different: if $\Theta'(E', k) = \{Es/xs\}V$ then $\Theta'(E', k@k') = \{Es/xs\}\Theta'(hd(V), k')$.

Particular cases of the isomorphism: We now analyze the diagram in Figure 1. 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. They are identified in Figure 1. In both cases, the fragment consists of the ν-nfs. 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 $\{hd(M)N/x\}P$. The latter are two decompositions of generalised elimination:

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

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

The λ-calculus is absent from Figure 1 (λ-terms form a subset of λgs), but there are three sets in bijective correspondence with the set of β-normal λ-terms, namely $\boxed{\text{Herbelin nfs}}$, $\boxed{\text{B}}$ and $\boxed{\text{Mints nfs}}$, the lower triangle. $\boxed{\text{Herbelin nfs}} \cong \boxed{\text{Mints nfs}}$ was known [4], the bijection being the restriction of ν to $\boxed{\text{Herbelin nfs}}$. A degenerate form of Theorem 2 is $\boxed{\text{Herbelin nfs}} \cong \boxed{\text{B}}$. The latter bijection (but not the former) extends to another bijection, namely $\boxed{\text{Schwichtenberg nfs}} \cong \boxed{\text{A}}$ (the former bijection does not extend to another bijection 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 [4, 19].

6 Final remarks

Contributions and related work: This paper completes Herbelin's programme, for the logic under analysis here. As compared to [11], we covered full sequent calculus, where the constraints on left introduction that define Herbelin's fragment LJT are dropped, but where the phenomenon of permutative conversions, typical of sequent calculus, shows up. In addition, we fully achieved the second goal of Herbelin's programme, residually present in [11], implicitly considered in [1] and already addressed in [5, 6]. The improvement over [1] and [5, 6] is that the spine calculus, when restricted to the logic of this paper, and the sequent systems in [5, 6] are all fragments of Herbelin's LJT and, therefore, are under the restrictions already mentioned.

In order to fully achieve the second goal, one has to define an extension of natural deduction that combines the idea of coercion calculus with von Plato's idea of generalised elimination rule [20]. On the one hand, von Plato's work goes much farther than this paper, in that [20] covers the whole language of first order logic; on the other hand, it lacks an analysis of the correspondence between cut-elimination and normalisation, indispensable to attaining the second goal. This

paper may then be seen as containing an extension of von Plato's work. 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: von Plato's normalisation, taken in the already slightly extended sense embodied in 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.

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 [10] and Prawitz [18] never could: elimination and substitution correspond to left introduction and cut, respectively. At the term calculi level, this mapping inverts the associativity of applicative terms, as envisaged by Herbelin. Then, such bijection at the level of proofs proves to be an isomorphism between cut-elimination and normalisation. This result improves, for the logic examined here, the classical results of Zucker and Pottinger [22, 17].

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. 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 symbolic 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 plan to keep using these languages in a more comprehensive study of permutative conversions. As the study of rule ν shows so far, calculus λ_{Nat} is no worse than calculus λ^{Gtz} for that purpose.

Conclusions: Herbelin's seminal suggestion in [11] 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. We proposed an abstract, robust extension of the concept of λ-term, under two concrete representations (λ^{Gtz}-terms and λ_{Nat}-terms), and studied the languages where these representations live. Representation questions (like whether there is direct head access in applicative terms) prove to have impact in the real word [1]. But, as expected, they also impact on foundational matters. Indeed, they allow a radical answer to a long-standing problem of structural proof-theory: if normalisation is extended as we propose, 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.

Acknowledgments: The author is supported by FCT, through Centro de Matemática, Universidade do Minho. We have used Paul Taylor's macros for typesetting Fig. 1.

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 International Conference on Functional Programming 2000*. IEEE, 2000.
3. R. Dyckhoff and L. Pinto. Cut-elimination and a permutation-free sequent calculus for intuitionistic logic. *Studia Logica*, 60:107–118, 1998.
4. R. Dyckhoff and L. Pinto. Permutability of proofs in intuitionistic sequent calculi. *Theoretical Computer Science*, 212:141–155, 1999.
5. 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/.
6. 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.
7. J. Espírito Santo. Delayed substitutions. In Franz Baader, editor, *Proceedings of RTA'07*, Lecture Notes in Computer Science. Springer-Verlag, 2007.
8. 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.
9. J. Gallier. Constructive logics. Part I: A tutorial on proof systems and typed λ-calculi. *Theoretical Computer Science*, 110:248–339, 1993.
10. G. Gentzen. Investigations into logical deduction. In M. E. Szabo, editor, *The collected papers of Gerhard Gentzen*. North Holland, 1969.
11. 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.
12. 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.
13. D. Kesner, L. Puel, and V. Tannen. A typed pattern calculus. *Information and Computation*, 124(1), 1995.
14. G. Mints. Normal forms for sequent derivations. In P. Odifreddi, editor, *Kreiseliana*, pages 469–492. A. K. Peters, Wellesley, Massachusetts, 1996.
15. S. Negri and J. von Plato. *Structural Proof Theory*. Cambridge, 2001.
16. M. Parigot. λμ-calculus: an algorithmic interpretation of classic natural deduction. In *Int. Conf. Logic Prog. Automated Reasoning*, volume 624 of *Lecture Notes in Computer Science*. Springer Verlag, 1992.
17. G. Pottinger. Normalization as a homomorphic image of cut-elimination. *Annals of Mathematical Logic*, 12:323–357, 1977.
18. D. Prawitz. *Natural Deduction. A Proof-Theoretical Study*. Almquist and Wiksell, Stockholm, 1965.
19. H. Schwichtenberg. Termination of permutative conversions in intuitionistic gentzen calculi. *Theoretical Computer Science*, 212, 1999.
20. J. von Plato. Natural deduction with general elimination rules. *Annals of Mathematical Logic*, 40(7):541–567, 2001.
21. P. Wadler. A Curry-Howard isomorphism for sequent calculus, 1993. Manuscript.
22. J. Zucker. The correspondence between cut-elimination and normalization. *Annals of Mathematical Logic*, 7:1–112, 1974.