Permutability of proofs in intuitionistic sequent calculi

Roy Dyckhoff*
School of Mathematical & Computational Sciences,
St Andrews University, St Andrews, Scotland

Luís Pinto*†
Departamento de Matemática,
Universidade do Minho, Braga, Portugal

Abstract

We prove a folklore theorem, that two derivations in a cut-free sequent calculus for intuitionistic propositional logic (based on Kleene's **G3**) are inter-permutable (using a set of basic "permutation reduction rules" derived from Kleene's work in 1952) iff they determine the same natural deduction. The basic rules form a confluent and weakly normalising rewriting system. We refer to Schwichtenberg's proof elsewhere that a modification of this system is strongly normalising.

Key words: intuitionistic logic, proof theory, natural deduction, sequent calculus.

1 Introduction

There is a folklore theorem that two intuitionistic sequent calculus derivations are "really the same" iff they are inter-permutable, using permutations as described by Kleene in [13]. Our purpose here is to make precise and prove such a "permutability theorem".

Prawitz [18] showed how intuitionistic sequent calculus derivations determine natural deductions, via a mapping φ from **LJ** to **NJ** (here we consider only the cut-free derivations and normal natural deductions respectively), and (in effect) that this mapping is surjective by constructing a right inverse of φ from **NJ** to **LJ**. Zucker [24] showed that, in the negative fragment of the calculus $\mathbf{LJ^c}$ (i.e. **LJ** including *cut*), two derivations have the same image under φ iff they are inter-convertible using a sequence of "permutative conversions", e.g. permutations of logical rules with the cut rule. In the present paper we prove a

*Supported by the European Commission via the ESPRIT BRA 7232 GENTZEN.
†Supported by the Centro de Matemática da Universidade do Minho, Braga, Portugal

similar result for a cut-free system, making precise the idea referred to above. In fact, we show how certain "permutation reduction rules" can be used to reduce an arbitrary derivation to "normal form" and that the set of such reductions is confluent. With minor changes this system is strongly normalising; we point to Schwichtenberg's [21] for a proof of this.

Our interest in these problems arises from the theory of logic programming, regarded as in [15] as based on proof search in a cut-free system; if one asks not just "What problems are solvable?" but "What solutions do these problems have?" and "How many times is each solution obtained?", one is led to analyse [5] the many-one relationship between sequent calculus derivations (suitable for proof search) and natural deductions (suitable for presenting solutions). In fact, Herbelin's sequent calculus (described below) is a much better basis for proof search than **LJ**, so the original problem disappears; nevertheless, in view of the historical importance of Gentzen's calculus [9] (and Kleene's variant [14] of it, **G3**) the permutability theorem is of independent interest.

Mints' paper [16] on the same topic came to our attention in October 1994, when an early version of this paper was being distributed; we discuss the relationship between his work and our own in §10. We thank Herbelin, Mints, Schwichtenberg and Troelstra for advance copies of their [10, 16, 21, 22] respectively. We are pleased to acknowledge that Herbelin's papers [10, 11] filled the gap between the usual definition of normal lambda-terms (representing natural deductions) and Prawitz' definition of ρ, our name for the right inverse of φ.

2 Background

2.1 Herbelin's calculus M

Herbelin [10, 11] gives a non-standard description (with origins in [2, 12, 20]) of terms representing normal natural deductions. Consider first a standard description of normal terms of the untyped lambda calculus:

$$\begin{array}{lcl} A & ::= & ap(A, N) \mid vr(V) \\ N & ::= & \lambda V.N \mid an(A) \end{array}$$

where **V** is a set of *variables*, **N** is the set of *normal terms* and **A** is the set of *application terms*. We use explicit constructors an and vr to ensure consistency with our type-checked implementations. The head variable of such a term is (for a large term) buried deep inside: Herbelin's representation brings it to the surface. So, following Herbelin (who calls the calculus $\overline{\lambda}$), we make the following:

Definition 1 *The set* **M** *of untyped deduction terms and the set* **Ms** *of lists of such terms are defined simultaneously as follows:*

$$\begin{array}{lcl} M & ::= & (V;Ms) \mid \lambda V.M \\ Ms & ::= & [] \mid M :: Ms \end{array}$$

Note the use again of the same symbol λ. The notation $[M_1, ..., M_n]$ abbreviates the term $M_1 :: ... :: M_n :: []$. The suggestion that such terms are lists is adequate while we deal with implication alone, but not when we add the other connectives. Terms are *equal* iff they are alpha-convertible; we use the symbol $\equiv$ for this relation.

Adding type restrictions gives us a description of the *typable* deduction terms. We call the associated typed system **MJ**, as it is intermediate between **LJ** and **NJ**, rather than use Herbelin's name **LJT** (already used in [4]).

There is a bijective translation between **M** and **N**, mentioned but not detailed in [11]: $(x;[M_1, ..., M_n])$ translates into the term $ap(...ap(x, N_1)..., N_n)$, usually written as $xN_1...N_n$, where N_i is the translation of M_i, and abstraction terms translate in the obvious way. The bijection extends to the typable terms: elsewhere [6, 7] we have called such sequent calculus *permutation-free*, meaning that there are no permutations, i.e. that the map from **MJ** to **NJ** is 1-1.

Further details of this calculus (covering all the connectives and several proofs of admissibility of cut) can be found in [10, 11, 6, 7]. We shall implicitly use the bijectiveness of the correspondences with **N** and **NJ** and not trouble to give proofs that (e.g.) a result shown for **M** translates correctly to a result claimed without proof for **N**.

2.2 The calculus LI

LI is a cut-free sequent calculus for intuitionistic implicational logic. First, *formulae* A are built up from *proposition variables* p, q, ... using just $\supset$ (for implication). Second, *contexts* Γ are finite sets of *variable : formula* pairs, associating at most one formula to each (term) variable in **V**. Third, there are *terms*, defined as in

Definition 2 *The set* **L** *of terms in cut-free* **LI** *derivations is defined as follows:*

$$L ::= var(V) \mid app(V, L, V.L) \mid \lambda V.L$$

The notions of *free* and *bound* variable and of *alpha-conversion* are as usual: there are two binding mechanisms, those at the occurrences of $V.L$ in the above definition. Two terms are said to be *equal* iff they are alpha-convertible; again, we shall use $\equiv$ for this relation. Note again the overloaded use of λ. We write $x \notin L$ for "x is not free in L"; similarly $x \in L$ for "x is free in L". Fourth, there are *judgments* $\Gamma \Rightarrow L : A$. Fifth, there are *typing rules*, inductively defining the derivations of the calculus:

$$\frac{}{x : A, \Gamma \Rightarrow var(x) : A}\ Axiom$$

$$\frac{y : A, \Gamma \Rightarrow L : B}{\Gamma \Rightarrow \lambda y.L : A \supset B}\ R\supset \qquad \frac{\Gamma \Rightarrow L_1 : A \quad y : B, \Gamma \Rightarrow L_2 : C}{\Gamma \Rightarrow app(x, L_1, y.L_2) : C}\ L\supset$$

with the provisos: $x : A \supset B$ belongs to Γ in $L\supset$; and y is *new*, i.e. does not appear in the context Γ, in both $L\supset$ and $R\supset$.

From the term and context parts of the end-sequent of a derivation, one can recover the entire derivation: the terms (modulo alpha conversion) are really just a convenient notation for derivations. The rules about new variables imply, for example, that bound variables are chosen so that the variable y in $app(x, L_1, y.L_2)$ differs from the variable x and does not occur (freely) in L_1. We make no distinction between the judgment $\Gamma \Rightarrow L : A$ and the assertion of its derivability.

Weakening is an admissible rule of **LI**: any derivation can be transformed to a weaker derivation by adding an assumption $x : A$ to each antecedent, for new x. The two derivations will be represented by the same term; also, if a derivation does not use an assumption $x : A$ then it can be strengthened by removing $x : A$ both from the end-sequent's antecedent and inductively (with descendants) from the premisses. In the following we use both the strengthening and the weakening techniques without comment.

2.3 The correspondence from L to M

Prawitz' description [18] (see also [23] §3.3.1) of the function φ from sequent calculus derivations to natural deductions uses the ordinary notion $[\cdot/\cdot]\cdot$ of substitution, recursively defined on the structure of the term being substituted into. Using Herbelin's definition of terms, we need a different version of the substitution function. This should be based on his cut rules, as in §9; for ease of exposition we now just introduce it in an *ad hoc* way. We do it just in the untyped case; typing is not necessary for the functions to be well-defined.

Definition 3 *The functions, of* substitution *of a variable x and a term M for a variable y in a term (resp. terms), are defined as follows:*

$$
\begin{array}{l}
subst : \mathbf{V} \times \mathbf{M} \times \mathbf{V} \times \mathbf{M} \longrightarrow \mathbf{M} \\
\quad \begin{array}{rcl}
subst(x, M, y, (y;Ms)) & =_{\mathsf{def}} & (x; M :: substs(x, M, y, Ms)) \\
subst(x, M, y, (z;Ms)) & =_{\mathsf{def}} & (z; substs(x, M, y, Ms)) \ \ (\text{if } \ z \neq y) \\
subst(x, M, y, \lambda z.M_1) & =_{\mathsf{def}} & \lambda z.subst(x, M, y, M_1)
\end{array}
\end{array}
$$

$$
\begin{array}{l}
substs : \mathbf{V} \times \mathbf{M} \times \mathbf{V} \times \mathbf{Ms} \longrightarrow \mathbf{Ms} \\
\begin{array}{rcl}
substs(x, M, y, []) & =_{\mathsf{def}} & [] \\
substs(x, M, y, M_1 :: Ms) & =_{\mathsf{def}} & subst(x, M, y, M_1) :: substs(x, M, y, Ms)
\end{array}
\end{array}
$$

Care is taken as usual to avoid variable capture, i.e. in line 3 of the definition for $subst$, $z \neq x$, $z \neq y$ and $z \notin M$.

Definition 4 *The function $\overline{\varphi} : \mathbf{L} \longrightarrow \mathbf{M}$ is defined as follows:*

$$
\begin{array}{rcl}
\overline{\varphi}(var(x)) & =_{\mathsf{def}} & (x;[]) \\
\overline{\varphi}(app(x, L_1, y.L_2)) & =_{\mathsf{def}} & subst(x, \overline{\varphi}L_1, y, \overline{\varphi}L_2) \\
\overline{\varphi}(\lambda x.L) & =_{\mathsf{def}} & \lambda x.\overline{\varphi}L
\end{array}
$$

Our definition is for untyped terms; we can easily extend it to typed terms and consider it as a map from cut-free sequent calculus derivations to normal natural deductions (in Herbelin's notation).

We say that L *determines* the term $\overline{\varphi}L$; and similarly for the derivation represented by L and the deduction represented by $\overline{\varphi}L$. We reserve the name φ (as in [24]) for the corresponding function (introduced but not named in [18], p. 91, REMARK) from **L** to **N**, defined by

$$\begin{array}{rcl} \varphi(var(x)) & =_{\mathsf{def}} & vr(x) \\ \varphi(app(x, L_1, y.L_2)) & =_{\mathsf{def}} & [ap(x, \varphi L_1)/y]\varphi L_2 \\ \varphi(\lambda x.L) & =_{\mathsf{def}} & \lambda x.\varphi L \end{array}$$

Note that φ is just the composite of $\overline{\varphi}$ with the bijection from **M** to **N**. Details are in [1].

Definition 5 *An equation $L_1 = L_2$ is φ-*trivial *iff $\varphi(L_1) \equiv \varphi(L_2)$; similarly for $\overline{\varphi}$-trivial, and similarly for permutations and transformations.*

2.4 The correspondence from M to L

Definition 6 *The function $\overline{\rho} : \mathbf{M} \longrightarrow \mathbf{L}$ is defined by recursion on the size of terms of* **M** *as follows:*

$$\begin{array}{rcl} \overline{\rho}(x;[]) & =_{\mathsf{def}} & var(x) \\ \overline{\rho}(x;M :: Ms) & =_{\mathsf{def}} & app(x, \overline{\rho}M, z.\overline{\rho}(z;Ms)) \ \ (z \text{ new}) \\ \overline{\rho}(\lambda x.M) & =_{\mathsf{def}} & \lambda x.\overline{\rho}M \end{array}$$

where $size(x;[M_1, ..., M_n]) = 1+\sum_{i=1}^{n} size(M_i)$ and $size(\lambda x.M) = 1+size(M)$.

Lemma 1 *$\overline{\varphi}(\overline{\rho}(M)) \equiv M$ for any M.* □

The definition is based on the construction in [18], which in fact described a right inverse to φ rather than to $\overline{\varphi}$. See §6.3 of [23] for a detailed account. Our definition is for untyped terms; we can easily extend it to typed terms and consider it as a map from normal natural deductions (in Herbelin's notation) to cut-free sequent calculus derivations.

3 Example

Consider the usual natural deduction (essentially the S combinator) of the sequent $A \supset (B \supset C), A \supset B, A \Rightarrow C$ in intuitionistic logic, where the two occurrences of A form an assumption class:

$$\dfrac{\dfrac{A \supset B \supset C \quad A}{B \supset C} \quad \dfrac{A \supset B \quad A}{B}}{C}$$

This deduction is represented, in the context $\langle z : A \supset (B \supset C), y : A \supset B, x : A\rangle$, by the term $ap(ap(vr(z), an(vr(x))), an(ap(vr(y), an(vr(x)))))$ of **N** and by the term $(z;[(x;[]), (y;[(x;[])])])$ of **M**.

Many different cut-free sequent calculus derivations determine this deduction: for example, those represented in the same context by the terms

$$\begin{array}{lll} S_1 & =_{\mathsf{def}} & app(z, x, w.app(w, app(y, x, v.v), u.u)) \\ S_2 & =_{\mathsf{def}} & app(z, x, w.app(w, app(y, x, v.v), u.app(y, x, v.u))) \\ S_3 & =_{\mathsf{def}} & app(z, x, w.app(y, x, v.app(w, v, u.u))) \\ S_4 & =_{\mathsf{def}} & app(z, app(y, x, v.x), w.app(y, x, v.app(w, v, u.u))) \\ S_5 & =_{\mathsf{def}} & app(y, x, v.app(z, x, w.app(w, v, u.u))) \end{array}$$

Commonly, these derivations are regarded as the same, because they are "permutation variants" of each other. The terms are related in the following ways, using the permutation reduction rules described in detail below:

$$S_5 \succ_{(ii)} S_4 \succ_{(i)} S_3 \succ_{(ii)} S_2 \succ_{(i)} S_1.$$

There are in fact infinitely many cut-free derivations with the same image $\varphi(S)$, by use of the permutation rule $\succ_{(i)}$ in reverse.

The purpose of this paper is to make such observations both precise and general. Kleene [13] discussed such permutations in the context of **LK** and **LJ**, without discussing the relationship with natural deductions. [23] gives a more detailed presentation of the theory of permutations.

4 Normality

In this section we give an intrinsic definition of the notion of normality for derivations, which will turn out to be equivalent both to irreducibility w.r.t. our permutation reduction rules and to being "canonical" as elements of the fibres of the mapping φ.

Definition 7 *Let L be a term of* $\mathbf{L}$. *L is* normal *iff in any subterm, of the form $app(x, L_1, y.L_2)$, L_2 is either $var(y)$ or of the form $app(y, L_3, z.L_4)$ with $y \notin L_3$ and $y \notin L_4$.*

Example: The term $S_1 =_{\mathsf{def}} app(z, x, w.app(w, app(y, x, v.v), u.u))$ of §3 is normal; the other terms in that section are not.

A normal term of the form $app(x_1, L_1, x_2.app(x_2, L_2, x_3.app(x_3, L_3, x_4.var(x_4))))$ is interpreted in $\mathbf{N}$ as $x_1 N_1 N_2 N_3$, where N_i interprets L_i; similarly for longer terms.

Lemma 2 (Normality Lemma) *For each term M of* $\mathbf{M}$, *$\overline{\rho}(M)$ is normal.*

Proof: By induction on the size of M.

Case M is $(x;[])$: then $\overline{\rho}(M)$ is just $var(x)$, which is normal.

Case M is $(x;M_1 :: Ms)$: then $\overline{\rho}(M)$ is $app(x, \overline{\rho}(M_1), z.\overline{\rho}(z;Ms))$ (new z); by induction $\overline{\rho}(M_1)$ and $\overline{\rho}(z;Ms)$ are normal. In fact, $\overline{\rho}(z;Ms)$ is either $var(z)$ or of the form $app(z, L_3, w.L_4)$ with z (since it was new) not free in L_3 or

L_4. Any application subterm of $\overline{\rho}(M)$ must be $\overline{\rho}(M)$ itself or a subterm either of $\overline{\rho}(M_1)$ or of $\overline{\rho}(z;Ms)$; in the first case, we have shown it has the desired form, in the second case we use the normality of $\overline{\rho}(M_1)$; in the third case we use the normality of $\overline{\rho}(z;Ms)$.

Case M is $\lambda x.M_1$: then $\overline{\rho}(M)$ is $\lambda x.\overline{\rho}(M_1)$; by induction $\overline{\rho}(M_1)$ is normal and obviously the abstraction of a normal term is normal. □

We will show the converse, that all normal terms L are of the form $\overline{\rho}(M)$. First, we identify a set of (permutation) reduction rules for reducing terms L to normal form.

5 Permutation reductions

Permutation reducibility is a relation between terms of **L**, formalised by means of the new judgment form $L_1 \succ L_2$, read as "L_1 and L_2 are terms of **L** and the first reduces to the second by a single permutation reduction". This relation is inductively generated by

$$\frac{L_1 \succ L_2}{\lambda x.L_1 \succ \lambda x.L_2}$$

$$\frac{L_1 \succ L_2}{app(x, L_1, y.L) \succ app(x, L_2, y.L)} \qquad \frac{L_1 \succ L_2}{app(x, L, y.L_1) \succ app(x, L, y.L_2)}$$

and the following "permutation reduction rules":

$$\begin{array}{ll}
\text{(i)} & app(x, L_1, y.L_2) \succ L_2 \;\; (\text{if } \; y \notin L_2) \\
\text{(ii)} & app(x, L_1, y.app(z, L_2, w.L_3)) \succ \\
 & \quad app(z, app(x, L_1, y.L_2), w.app(x, L_1, y.L_3)) \;\; (\text{if } \; y \neq z) \\
\text{(ii}') & app(x, L_1, y.app(y, L_2, w.L_3)) \succ \\
 & \quad app(x, L_1, y'.app(y', app(x, L_1, y.L_2), w.app(x, L_1, y.L_3))) \\
\text{(iii)} & app(x, L_1, y.\lambda z.L_2) \succ \lambda z.app(x, L_1, y.L_2)
\end{array}$$

with the constraint in (ii$'$) that y' is new, and the constraints that, in (ii) and (ii$'$), y is free in L_2 or in L_3, since otherwise $app(z, L_2, w.L_3)$ in the LHS of (ii) matches L_2 in the LHS of (i) or (respectively) the RHS of (ii$'$) reduces by (i) back to the LHS.

Note: (i) and (ii) may be combined (when $y \neq z$ and $y \notin L_2$ but $y \in L_3$) to yield the elegant permutation:

$$\text{(v)} \;\; app(x, L_1, y.app(z, L_2, w.L_3)) \succ app(z, L_2, w.app(x, L_1, y.L_3))$$

(The LHS reduces by (ii) to $app(z, app(x, L_1, y.L_2), w.app(x, L_1, y.L_3))$, which reduces by (i) to the RHS. Note that scope rules for the LHS imply that $w \neq x$ and $w \notin L_1$, so, if $w \in L_3$, (v) can be used again (and again...).)

Note: We could also use the rule

$$\text{(iv)} \;\; app(x, L_1, y.L_2) \succ app(x, L_1, y.app(x, L_1, z.[z/y]L_2))$$

where z is new and $|z/y|L_2$ indicates L_2 in which zero or more occurrences of y are replaced by z. Using (iv), (ii), (i) and (ii) we obtain (ii′).

Although (iv) seems more primitive, our main theorem is most naturally proved using (ii′) (and establishes by induction that instances of (iv) are obtainable using (i), (ii), (ii′) and (iii)).

From now on, we use the symbol $\succ$ for the permutation reducibility relation and $\prec$ for its transpose. $\succ^*$ and $\prec^*$ denote as usual the reflexive transitive closures of the relations $\succ$ and $\prec$. $\approx$ denotes the reflexive symmetric transitive closure of $\succ$. We say that L_1 and L_2 are *interpermutable* when $L_1 \approx L_2$. We say that L_1 *reduces** to L_2 (or that L_1 is *reducible** to L_2) iff $L_1 \succ^* L_2$.

Rule (i) simplifies the derivation by removing an unnecessary step; (ii) permutes instances of $L\supset$ past each other, as in [13]; (ii′) (roughly) achieves the effect of (ii) when one principal formula originates in the other; (iii) permutes $L\supset$ past $R\supset$, as in [13]. Rules (i) and (ii′) are not "permutations" in Kleene's sense, because the principal formula of the top rule occurs as an active formula of the lower rule. Kleene however allowed structural rules, of which we have none. Rules (i) and (iv) (from which (ii′) can be derived) correspond to his modification of derivations with structural rules.

Proposition 1 *Each of these permutation reduction rules is φ- (and $\overline{\varphi}$-) trivial.*

Proof: Routine: consider, for example, (ii′), with (y' new)

$$\begin{array}{ll} & \varphi(app(x, L_1, y.app(y, L_2, w.L_3))) \\ = & [ap(x, \varphi(L_1))/y][ap(y, \varphi(L_2))/w]\varphi(L_3) \\ = & [ap(ap(x, \varphi(L_1)), [ap(x, \varphi(L_1))/y]\varphi(L_2))/w][ap(x, \varphi(L_1))/y]\varphi(L_3) \\ = & [ap(x, \varphi(L_1))/y'][ap(y', [ap(x, \varphi(L_1))/y]\varphi(L_2))/w][ap(x, \varphi(L_1))/y]\varphi(L_3) \\ = & \varphi(app(x, L_1, y'.app(y', app(x, L_1, y.L_2), w.app(x, L_1, y.L_3)))). \ \Box \end{array}$$

We shall see in §9 examples of permutation rules from [13] that involve disjunction and are not φ-trivial.

6 Irreducibility

Here we show that normal terms are irreducible; later we show the converse.

Definition 8 *L is* irreducible *iff no reduction is applicable to L.*

Lemma 3 (Irreducibility Lemma) *Each normal term L is irreducible.*

Proof: Since subterms of normal terms are normal, we need only check, for each rule, normal instances L of the LHS. We consider the cases in turn:

Rule (i): L is of the form $app(x, L_1, y.L_2)$ for $y \notin L_2$. By normality, L_2 is either $var(y)$ or $app(y, L_3, z.L_4)$, contrary to $y \notin L_2$.

Rule (ii): L is of the form $app(z, L_1, y.app(z, L_2, w.L_3))$ for $y \neq z$. By normality, $y = z$, a contradiction.

Rule (ii′): L is of the form $app(z, L_1, y.app(y, L_2, w.L_3))$ with y free in L_2 or L_3. By normality, y is not free in L_2 or L_3, a contradiction.

Rule (iii): L is of the form $app(z, L_1, y.\lambda z.L_2)$. By normality, $\lambda z.L_2$ must be $var(y)$ or an application, which are impossible. □

7 Normalisability

The argument here is based on Herbelin's calculus, to make the induction easier. One might also use the description $\lambda\vec{x}.(...((xN_1)N_2)...N_n)$ of normal terms; but this description is not so convenient in a mechanical verification [1] and it is not easy to handle connectives such as disjunction.

Lemma 4 (Permutability Lemma) *Let M_1 and M_2 be terms of* **M**. *Then*

$$app(x, \overline{\rho}M_1, y.\overline{\rho}M_2) \succ^* \overline{\rho}(subst(x, M_1, y, M_2)).$$

Proof: By induction on the size of M_2. When y is not free in $\overline{\rho}M_2$, the LHS reduces by permutation (i) to $\overline{\rho}M_2$, to which the RHS is identical by simplification; so we may assume that $y \in \overline{\rho}M_2$.

Case 0: $size(M_2) = 1$, so M_2 is $(z;[])$ for some variable z, which by our assumption must be y. So the LHS is $app(x, \overline{\rho}M_1, y.\overline{\rho}(y;[]))$

$$\begin{array}{lll}
\equiv & app(x, \overline{\rho}M_1, y.var(y)) & \text{(by definition of } \overline{\rho}) \\
\equiv & \overline{\rho}(x;[M_1]) & \text{(by definition of } \overline{\rho}) \\
\equiv & \overline{\rho}(subst(x, M_1, y, (y;[]))) & \text{(by definition of } subst)
\end{array}$$

which is the RHS. So, in this case the LHS and the RHS are identical.

Case 1: $size(M_1) > 1$; we suppose the lemma is true for all M_2 of lesser size. Then, M_2 is either of the form $(z;M :: Ms)$ or of the form $\lambda z.M$, and in the former case, two subcases arise according to whether $z \neq y$ or $z = y$:

Case 1(ii): $M_2 \equiv (z;M :: Ms)$, when $z \neq y$; by assumption, y is free in $M :: Ms$. So the LHS is $app(x, \overline{\rho}M_1, y.\overline{\rho}(z;M :: Ms))$

$$\begin{array}{ll}
\equiv & app(x, \overline{\rho}M_1, y.app(z, \overline{\rho}M, z_1.\overline{\rho}(z_1;Ms))) \\
 & \text{(by definition of } \overline{\rho}\text{, where } z_1 \text{ is new)} \\
\succ & app(z, app(x, \overline{\rho}M_1, y.\overline{\rho}M), z_1.app(x, \overline{\rho}M_1, y.\overline{\rho}(z_1;Ms))) \\
 & \text{(by permutation reduction rule (ii))} \\
\succ^* & app(z, \overline{\rho}(subst(x, M_1, y, M)), z_1.app(x, \overline{\rho}M_1, y.\overline{\rho}(z_1;Ms))) \\
 & \text{(by induction, since } size(M) < size(z;M :: Ms)) \\
\succ^* & app(z, \overline{\rho}(subst(x, M_1, y, M)), z_1.\overline{\rho}(subst(x, M_1, y, (z_1;Ms)))) \\
 & \text{(by induction, since } size(z_1;Ms) < size(z;M :: Ms)) \\
\equiv & app(z, \overline{\rho}(subst(x, M_1, y, M)), z_1.\overline{\rho}(z_1;substs(x, M_1, y, Ms))) \\
 & \text{(by definition of } \overline{\rho}\text{, using } z_1 \neq y)
\end{array}$$

$$\begin{aligned} &\equiv \overline{\rho}(z; subst(x, M_1, y, M) :: substs(x, M_1, y, Ms)) \\ &\quad \text{(by definition of } subst\text{, since } z_1 \text{ is new)} \\ &\equiv \overline{\rho}(subst(x, M_1, y, (z; M :: Ms))) \\ &\quad \text{(by definition of } subst\text{, since } z \neq y\text{)} \end{aligned}$$

which is the RHS.

Case 1(ii′): $M_2 \equiv (y;M :: Ms)$: Two subcases arise: y free in $M :: Ms$ and otherwise. The first subcase is routine, similar to 1(ii) but using rule (ii′). In the second subcase, where y is not free in $M :: Ms$, by direct computation,

$$app(x, \overline{\rho}M_1, y.\overline{\rho}(y;M :: Ms)) \equiv \overline{\rho}(subst(x, M_1, y, (y;M :: Ms))).$$

Case 1(iii): $M_2 \equiv \lambda z.M$; routine, using rule (iii). □

Theorem 1 *For every term L of* **L**, $L \succ^* \overline{\rho}(\overline{\varphi}(L))$.

Proof: By induction on the structure of L. First, suppose L is a variable x; then (trivially) the LHS and RHS are identical, using the definitions of $\overline{\varphi}$ and $\overline{\rho}$. Second, the case when $L \equiv \lambda x.L_1$ is a routine use of the induction hypothesis. Third, if $L \equiv app(x, L_1, y.L_2)$, then L is (by induction, twice) reducible* to $app(x, \overline{\rho}(\overline{\varphi}(L_1)), y.\overline{\rho}(\overline{\varphi}(L_2)))$ and by the permutability lemma this reduces* to $\overline{\rho}(subst(x, \overline{\varphi}(L_1), y, \overline{\varphi}(L_2)))$, i.e. to $\overline{\rho}(\overline{\varphi}(app(x, L_1, y.L_2)))$, which is just $\overline{\rho}(\overline{\varphi}(L))$. □

Corollary 1 *For every term L of* **L**, *$L \succ^* \rho(\varphi(L))$; and for every pair L_1, L_2 of terms of* **L**, *(i) $\overline{\varphi}(L_1) \equiv \overline{\varphi}(L_2)$ iff $L_1 \approx L_2$ and (ii) $\varphi(L_1) \equiv \varphi(L_2)$ iff $L_1 \approx L_2$.*

Proof: (i) $\overline{\varphi}(L_1) \equiv \overline{\varphi}(L_2)$ implies that $L_1 \succ^* \overline{\rho}(\overline{\varphi}(L_1)) \equiv \overline{\rho}(\overline{\varphi}(L_2)) \prec^* L_2$; the converse follows by Proposition 1. □

Theorem 2 *Let L be a term of* **L**. *The following are equivalent:*

1. *L is normal;*
2. *L is irreducible;*
3. *$L \equiv \overline{\rho}(\overline{\varphi}(L))$;*
4. *L is of the form $\overline{\rho}(M)$ for some M.*

Proof: (1)⇒(2) follows by the irreducibility lemma (3); (2)⇒(3) is from theorem 1; (3)⇒(4) is trivial; (4)⇒(1) follows by the normality lemma (2). □

Thus theorem 1 is a weak normalisability result; every term L can be reduced* to a normal form (and the normal forms are the irreducible terms).

8 Confluence and Strong Normalisation

Theorem 3 *The rewriting system (i), (ii), (ii′), (iii) is confluent on* **L**.

Proof: Suppose $L \succ^* L_1$ and $L \succ^* L_2$. Then $\varphi(L) \equiv \varphi(L_1) \equiv \varphi(L_2)$, since the reductions are φ-trivial. So all of L, L_1 and L_2 reduce* to the same normal form, $\rho(\varphi(L))$. □

Without further restrictions, the system of rules is non-terminating: e.g. rule (v) can be used repeatedly, and (v) depends on (unrestricted) (ii) and (i). Note that (ii) can be used repeatedly on its own, because e.g. (assuming $y \neq z$, $w \in L_3$ and $y \in L_3$)

$$\begin{array}{l} app(x, L_1, y.app(z, L_2, w.L_3)) \ \succ \\ app(z, app(x, L_1, y.L_2), w.app(x, L_1, y.L_3)) \ \succ \\ app(x, app(z, app(x, L_1, y.L_2), w.L_1), y.app(z, app(x, L_1, y.|y/y|L_2), w.L_3)) \succ \ldots \end{array}$$

where the second reduction is allowed because $x \neq y$ (implicitly, because of the scoping rules). To restrict this, while at the same time allowing enough reductions for the proof of the permutability lemma to work, is tricky.

The instances of the permutation reduction rules used in the proof have their L arguments of the form $\overline{\rho}M$, which we saw in Theorem 2 to be exactly the normal terms. Thus the proof of the lemma incorporates an innermost reduction strategy; this suggests one should conjecture that the system is strongly normalising if one makes restrictions such as normality of the arguments of terms being reduced. Let x be a variable; we say that a term L is *x-normal* iff L is either $var(x)$ or is $app(x, L_1, y.L_2)$ with $x \notin L_1$ and $x \notin L_2$ and L_2 being y-normal. Clearly terms of the form $\overline{\rho}(z;Ms)$ are z-normal for $z \notin Ms$.

Conjecture 1 *The rewriting system (i), (ii), (ii′), (iii) is SN if*

(a) rules (ii), (ii′) are restricted to cases where the argument L_3 of the LHS $app(x, L_1, y.app(z, L_2, w.L_3))$ is w-normal; and

(b) rules (ii), (ii′) are restricted to cases where the arguments L_1, L_2 and L_3 of the LHS $app(x, L_1, y.app(z, L_2, w.L_3))$ are normal.

Note that with these restrictions, the proof of the permutability lemma still works.

Schwichtenberg [21] outlines a proof of this conjecture, strengthened by omission of condition (b), as follows. He develops a new notation, *binary sequent terms*, in which $M_v\{y, L\}$ corresponds to our $app(y, L, v.M)$, hinting at the translation $\hat{\cdot}$ to natural deduction terms $\hat{M}_v[y\hat{L}]$ (which we would write as $[ap(y, \varphi L)/v]\varphi(M)$). More generally there are *multiary sequent terms* such as $M_v\{y, L_1 L_2\}$ corresponding to our $app(y, L_1, w.app(w, L_2, v.M))$ (where $w \notin L_2$ and $w \notin M$), and similarly for vectors $\vec{L}$ of terms in place of $L_1 L_2$. Our rule (ii) (restricted by condition (a) and with, for ease of exposition, a very restricted form $app(w, N, w_1.w_1)$ of the argument L_3) is translated to the reduction (6')

$$(w_1)_{w_1}\{w, N\}_w\{z, L_2\}_y\{x, L_1\} \to (w_1)_{w_1}\{w, N\}_y\{x, L_1\}_w\{z, (L_2)_y\{x, L_1\}\}$$

in which N, L_1 and L_2 may in fact be vectors (and thus $(w_1)_{w_1}\{w, \vec{N}\}$ represents the general form of the L_3 argument allowed by the strengthened form of the conjecture. The other rules are represented similarly, e.g. (ii$'$) by (7'). For example, our reduction by (ii) of S_5 to S_4 (from §3) is simulated by the reduction

$$u_u\{w, v\}_w\{z, x\}_v\{y, x\} \to u_u\{w, v\}_v\{y, x\}_w\{z, x_v\{y, x\}\}.$$

Termination of the rule set { (1), (5) (6'), (7') } (and of some similar rule sets) is shown in [21] using a decreasing measure δ on terms. The termination of our rule set {(i), (ii), (ii$'$), (iii)} (with the restrictions mentioned above) therefore follows, thus establishing the strengthened version of our conjecture. It would be of interest to have a full and direct proof of this without using the multiary notation (on which the measure function depends) of [21].

9 Extension to other logical constants

This section considers the extension of the theory to cover the other intuitionistic logical constants. We refer to the full paper [8] for details. The main point of interest is that some of the Kleene-style permutations [13] are not φ-trivial.

Kleene's analysis was for a system with primitive structural rules. We can consider the following table, in which the intersection of the row R and the column C refers to the permutable pair R/C in which R lies above C and may be permuted to below it:

	$L\supset$	$R\supset$	$L\wedge$	$R\wedge$	$L\vee$	$R\vee$
$L\supset$	••	X	••	–	X	–
$R\supset$	•	XX	•	XX	X	XX
$L\wedge$	••	–	••	–	–	–
$R\wedge$	•	XX	•	XX	X	XX
$L\vee$	••	N	••	N	N	N
$R\vee$	•	XX	•	XX	X	XX

In this table: • indicates that there is a single permutation reduction rule; •• indicates that there is a pair of reduction rules; – indicates that there is a permutable pair but it is not used in the proof of the permutability theorem, because it is the reverse of a permutable pair that is used; X indicates that the permutation is forbidden; XX that there is no permutable pair because both R and C are right rules; and N indicates that the permutation is not φ-trivial, essentially because the notion of normality used in **NJ** does not allow introduction rules to be permuted up into minor premisses of elimination rules. Each permutation that is marked N in the table, e.g.

$$L\vee/R\supset \qquad \lambda x.when(y, z_1.L_1, z_2.L_2) \approx when(y, z_1.\lambda x.L_1, z_2.\lambda x.L_2) \quad x \neq y$$

(using the notation of [8]), is not φ-trivial; if we apply φ to the two sides of $L\vee/R\supset$, then we get normal terms representing $\supset I$- and $\vee E$-steps respectively.

10 Related work

Theorem 1 of §4 of [24], for the negative fragment of intuitionistic logic, is similar to (ii) of our corollary 1, but for the systems with cut. Zucker's argument, showing that two derivations with the same image under φ are interpermutable, is a case analysis on the last steps of the two derivations; for example, the case of both last steps being $L\supset$ is dealt with by use of derivations with cut. Thus his notion of "interpermutable" uses permutations involving the cut rule. (Moreover, there is no reference in [24] to Kleene's theory of permutations.) See [17] for further discussion (but still for the systems with cut) of Zucker's results.

Mints [16] (available to us after our own proof of an early version of theorem 1, using $\approx$ rather than $\succ^*$) proves the same theorem (but without clarifying whether or not the permutations are directed and which permutations are required) by means of an induction on the structure of derivations, in the general case (not just propositional logic); our use of the term notation for derivations allows, in contrast, the nature of the permutations to be made precise and amenable to mechanical treatment [1]. His work applies to Gentzen's system **LJ** with explicit weakening and contraction rules rather than, as in our case, to Kleene's G3, where these rules are built into the logical rules. Our (iv) corresponds to his use of transformations to move contraction; similarly, our (i) corresponds to his transformations to move weakening down towards the root. He also describes the normal forms using constraints on the structure of derivations, similar to ours.

Troelstra [22] has proved a similar weak normalisation theorem for a Gentzen calculus based on **G3i** [23], with the normal derivations being in 1-1 correspondence with natural deductions in long normal form under the complete discharge convention. This calculus lacks the term labels that we have used both to facilitate the naming of derivations (and their permutations) and because of the connections with logic programming viewed as a search for normal terms inhabiting formulae viewed as types. [22] also mentions some difficulties in Mints' treatment of contraction.

Bellin and van de Wiele [3] prove a similar result for a multiplicative linear logic without propositional constants, relating sequent calculus derivations to proof nets. Andreoli's work [2] on focusing proofs in linear logic seems to be related, in its stringent normality conditions on proofs; but there is no permutability theorem (yet). Pym and Wallen [19] prove a theorem (5.7), showing how any derivation (maybe ill-typed) of the $\lambda\Pi$-calculus can be permuted to obtain a (well-typed) derivation.

Schwichtenberg [21] develops a new notation, *multiary sequent terms*, representing derivations of **LJ**, a notion of *multiary normal form*, permutative conversions and a measure function with respect to which the conversion rules are decreasing. Our §8 discusses the use of this theory to prove a result about strong termination for our rules.

11 Conclusion

We have made precise, for intuitionistic propositional logic, the idea that two proofs are really the same iff they are interpermutable; moreover, we have presented a rewriting system, confluent and weakly normalising, for reduction of terms (representing cut-free sequent calculus derivations) to normal form. That this can be made SN by appropriate restrictions (for the implicational fragment) follows from Schwichtenberg's results in [21]. For all the propositional connectives, we have identified precisely which of the Kleene-style permutations are required (and pointed out some that are inappropriate). Our methods illustrate the utility of Herbelin's representation of lambda-terms which brings the head variable to the outside. We are confident that the methods generalise to first-order logic: see [8] and its successors for details in due course.

References

[1] A. Adams, *Tools and techniques for machine-assisted meta-theory*, (PhD thesis, Univ. St Andrews, 1997).

[2] J. Andreoli, Logic programming with focusing proofs in linear logic, *J. Logic & Computation* **2** (1992) 297–347.

[3] G. Bellin and J. van de Wiele, Empires and kingdoms in MLL^-, in: J.-Y. Girard, Y. Lafont and L. Regnier, eds., *Advances in Linear Logic* (Cambridge University Press, 1995) 249–270.

[4] R. Dyckhoff, Contraction-free sequent calculi for intuitionistic logic, *J. of Symbolic Logic* **57** (1992) 795–807.

[5] R. Dyckhoff and L. Pinto, Uniform proofs and natural deductions, in: D. Galmiche and L. Wallen, eds., *Proceedings of CADE-12 workshop on "Proof search in type theoretic languages"* (Nancy, 1994) 17–23.

[6] R. Dyckhoff and L. Pinto, *A permutation-free sequent calculus for intuitionistic logic*, Univ. St Andrews Comp. Sci. Research Report CS/96/9 (August 1996), from "http://www-theory.dcs.st-and.ac.uk/~rd/".

[7] R. Dyckhoff and L. Pinto, Cut-elimination and a permutation-free sequent calculus for intuitionistic logic, *Studia Logica*, to appear (submitted 1996).

[8] R. Dyckhoff and L. Pinto, *Permutability of inferences in intuitionistic sequent calculi*, Univ. St Andrews Comp. Sci. Research Report CS/97/7 (June 1997), from "http://www-theory.dcs.st-and.ac.uk/~rd/".

[9] G. Gentzen, *The collected papers of Gerhard Gentzen* (ed. M. Szabo, North-Holland, Amsterdam, 1969).

[10] H. Herbelin, *A λ-calculus structure isomorphic to sequent calculus structure*, pre-print (October 1994); now available at "http://capella.ibp.fr/~herbelin/LAMBDA-BAR-FULL.dvi.gz".

[11] H. Herbelin, A λ-calculus structure isomorphic to Gentzen-style sequent calculus structure, in: L. Pacholski and J. Tiuryn, eds., *Proc. of the Conf. on Computer Science Logic* (Kazimierz, 1994), Springer LNCS **933**, 61–75.

[12] W. Howard, The formulae-as-types notion of construction, in: R. Hindley and J. Seldin, eds., *Curry Festschrift* (Academic Press, 1980) 479–490.

[13] S. Kleene, Permutability of inferences in Gentzen's calculi LK and LJ, *Mem. Amer. Math. Soc.* (1952) 1–26.

[14] S. Kleene, *Introduction to metamathematics* (Walters-Noordhoff & North-Holland, 1991).

[15] D. Miller, G. Nadathur, F. Pfenning and A. Scedrov, Uniform proofs as a foundation for logic programming, *Annals of Pure and Applied Logic* **51** (1991) 125–157.

[16] G. Mints, Normal forms of sequent derivations, in: P. Odifreddi, ed., *Kreiseliana* (A. K. Peters, Wellesley, Massachusetts, 1996) 469–492; also part of Stanford Univ. Report CSLI-94-193 (November 1994).

[17] G. Pottinger, Normalization as a homomorphic image of cut-elimination, *Annals of Mathematical Logic* **12** (1977) 323–357.

[18] D. Prawitz, *Natural deduction* (Almquist & Wiksell, Stockholm, 1965).

[19] D. Pym and L. Wallen, Proof search in the $\lambda\Pi$-calculus, in: G. Huet and G. Plotkin, eds., *Logical frameworks* (Cambridge University Press, 1991) 309–340.

[20] H. Schellinx, *The noble art of linear decorating* (PhD thesis, Univ. Amsterdam, 1994).

[21] H. Schwichtenberg, Termination of permutative conversions in intuitionistic Gentzen calculi (*Theoretical Computer Science*, this issue).

[22] A. S. Troelstra, Marginalia on sequent calculi, (*Studia Logica*, to appear); also University of Amsterdam, Inst. for Logic, Language and Computation, Research Report ML-1998-01 (January 1998).

[23] A. S. Troelstra and H. Schwichtenberg, *Basic proof theory*, (Cambridge University Press, 1996).

[24] J. Zucker, The correspondence between cut-elimination and normalization, *Annals of Mathematical Logic* **7** (1974) 1–112.