Strong cut-elimination systems for Hudelmaier's depth-bounded sequent calculus for implicational logic

Roy Dyckhoff[1], Delia Kesner[2], and Stéphane Lengrand[1,2]

[1] School of Computer Science, University of St Andrews, Scotland
[2] PPS, CNRS and Université Paris 7, France

Abstract. Inspired by the Curry-Howard correspondence, we study *normalisation* procedures in the depth-bounded intuitionistic sequent calculus of Hudelmaier (1988) for the implicational case, thus strengthening existing approaches to *Cut*-admissibility. We decorate proofs with terms and introduce various term-reduction systems representing proof transformations. In contrast to previous papers which gave different arguments for *Cut*-admissibility suggesting *weakly normalising* procedures for *Cut*-elimination, our main reduction system and all its variations are *strongly normalising*, with the variations corresponding to different optimisations, some of them with good properties such as confluence.

1 Introduction

The sequent calculus **G4ip** (as it is called in [TS00]) for intuitionistic propositional logic was independently developed in [Hud89,Hud92] and [Dyc92]; see also [LSS91]; it has the strong property of being *depth-bounded*, in that proofs are of bounded depth and thus (for root-first proof search) no loop-checking is required. This contrasts with other calculi for this logic such as Kleene's **G3ip**, where proofs can be of unbounded depth. Its essential ingredients appeared already in 1952 work of Vorob'ev, published in detail in [Vor70].

Its completeness can be shown by various means, either indirectly, using the completeness of another calculus and a permutation argument [Dyc92], or directly, such as in [DN00] where cut-admissibility is proved without reference to the completeness of any other sequent calculus. This admissibility proof could be seen, via the Curry-Howard correspondence, as a *weakly normalising* proof-reduction system. Developing this idea, this paper presents a formulation of implicational **G4ip** with derivations represented by terms; strong (instead of weak) normalisation is proved by the use of a multi-set path ordering. Several variations, *all of them being strongly normalising*, are considered.

The merits of **G4ip** for proof-search and automated reasoning have been discussed in many papers (see [ORK05] for some recent pointers; note its use of an old name **LJT** for **G4ip**). However, a question that has been less investigated is the following: what are the proofs expressed in **G4ip** and what is their semantics ? Here we investigate operational, rather than denotational, semantics because it is more directly related to inductive proofs of cut-admissibility (such as in [DN00]). Further work will investigate denotational semantics, by relating these proofs and their reductions to the simply-typed λ-calculus.

This paper presents **G4ip** with a term syntax, so sequents are of the form $\Gamma \Rightarrow M\!:\!A$ where A is a type, M is a term and Γ is a consistent finite set of "declarations" of the form $x\!:\!B$, where x is a variable and B a type. Results about such sequents translate directly to results about traditional "logical sequents".

Our approach to cut-elimination using terms differs from that in [DN00], which showed (using logical sequents) first the admissibility of *Contraction* and then the admissibility of "context-splitting" (or "multiplicative") *Cut*. Given our interest in term calculi, it is appropriate to use rather a "context-sharing" (or "additive") *Cut*; admissibility of *Contraction* then follows as a special case of that of *Cut*.

To some extent, Matthes [Mat02] also investigated terms and reductions corresponding to cut-elimination in **G4ip**, with a variety of motivations, such as that of understanding better Pitts' algorithm [Pit92] for uniform interpolation. His work is similar to ours in using terms to represent derivations; but it differs conceptually from ours by considering not the use of explicit operators for the *Cut*-rule but the closure of the syntax under (implicit) substitution, as in pure λ-calculus, where the general syntax of λ-terms may be regarded as the extension of the normal λ-terms by such a closure. His reduction rules are global (using implicit substitutions) rather than local (using explicit operators); strong normalisation is shown for a subset of the reductions, but unfortunately not for all that are required.

Structure of the paper The paper is organised as follows. Section 2 presents the term syntax and typing rules of our calculus for **G4ip** and its auxiliary (admissible) rules. Section 3 studies proof transformations and reduction rules of the calculus. Section 4 shows a translation from the calculus to a first-order syntax and Section 5 shows that every reduction step satisfies subject reduction and decreases first-order terms associated to derivations with respect to a multi-set path ordering, thus proving strong normalisation. In Section 6 we give different variants for the reduction system introduced in Section 3, some of them being confluent. Finally we conclude and give some ideas for further work. We refer the reader to the full version [DKL06] of this paper for further details such as complete proofs.

2 Syntax

2.1 Grammar

We assume we are given an infinite set of *base types* P (known as *proposition variables* or *atomic formulae* in the logical interpretation) and an infinite set of *variables* x. We consider the following grammars for *types* (also known as *formulae*) and *terms*:

Definition 1 (Grammar of Types and Terms).

$$
\begin{array}{rl}
A, B & ::= P \mid A \supset B \\
M, N & ::= x \mid \lambda x.M \mid x(y, z.M) \mid x(u.v.M, z.N) \mid \\
 & \quad \mathtt{inv}(x, y.M) \mid \mathtt{of}(M, x) \mid \mathtt{dec}(x, y, z.M) \mid \mathtt{cut}(M, x.N)
\end{array}
$$

In this definition, the first line defines the syntax for types, the second gives the syntax for *normal* or *constructor* terms (corresponding to primitive derivations) and the third gives the extra syntax for *auxiliary* terms, which may be built up using also the "auxiliary constructors" that appear in bold teletype font, such as `cut`. Six of the eight term constructors use variable binding: in $\lambda x.M$, x binds in M; in $x(y, z.M)$, z binds in M; in $x(u.v.M, z.N)$, u and v bind in M and z binds in N; in $\mathtt{inv}(x, y.M)$, y binds in M; in $\mathtt{dec}(x, y, z.M)$, z binds in M; and in $\mathtt{cut}(M, x.N)$, x binds in N. Lack of space here does not allow a formal treatment of variable binding using e.g. De Bruijn indices or nominal logic [Pit03].

Barendregt's convention is used to avoid confusion of free and bound variables, and α-convertible terms are, as usual, regarded as identical.

Certain constraints on the use of the term syntax will be evident once we present the typing rules; these constraints are captured by the following notion of *well-formed term*:

Definition 2. *A term L is* well-formed *if in any sub-term of the form*

- *$x(y, z.M)$, we have $x \neq y$, with x not free in M;*
- *$x(u.v.M, z.N)$, we have $u \neq v$, with x not free in M and not free in N;*
- *$\mathtt{inv}(x, y.M)$, we have x not free in M;*
- *$\mathtt{of}(M, x)$, we have x not free in M;*
- *$\mathtt{dec}(x, y, z.M)$, we have $x \neq y$, with both of them not free in M.*

Definition 3 (Ordering on (multi-sets of) types). *The* weight $w(A)$ *of a type A is defined by: $w(P) = 1$ for any base type P and $w(A \supset B) = 1 + w(A) + w(B)$. Types are compared by their weight, i.e. we say that A is* smaller *than B iff $w(A) < w(B)$.*

We shall then compare multi-sets of types, equipped with the traditional multi-set ordering *[DM79,BN98], denoted $<_{\mathsf{mul}}$, generated by the order relation on types.*

The weight is chosen to ensure that, for every rule of the logical sequent calculus **G4ip**, the multi-set of types appearing in the conclusion is greater than that of each premiss. Hence, we say that **G4ip** is *depth-bounded*. See [Dyc92] or [TS00] for details, and see the next section for the corresponding property in our version of **G4ip** with terms.

2.2 Typing

A *context* Γ is a finite mapping from variables to types. The variable x is *declared* in Γ when $\Gamma(x)$ is defined. When we write a context in the form $\Gamma, x\colon A$ (i.e. the extension of Γ with $x \mapsto A$), it is always implicit that x is not declared in Γ. We denote by $m(\Gamma)$ the range (considered as a multi-set) of a context Γ.

A *sequent* consists of a context Γ, a term M and a type A; it is written $\Gamma \Rightarrow M : A$.

The next definition adds term notation to the rules for implication of **G4ip**; another view is that it shows how the untyped normal terms of the above grammar may be typed.

Definition 4 (Typing Rules for Normal Terms).

$$\frac{}{\Gamma, x : A \Rightarrow x : A}\,Ax \qquad \frac{\Gamma, x : A \Rightarrow M : B}{\Gamma \Rightarrow \lambda x.M : A \supset B}\,R\supset$$

$$\frac{\Gamma, y : A, z : B \Rightarrow M : E}{\Gamma, x : A \supset B, y : A \Rightarrow x(y, z.M) : E}\,L0\supset$$

$$\frac{\Gamma, u : C, v : D \supset B \Rightarrow M : D \qquad \Gamma, z : B \Rightarrow N : E}{\Gamma, x : (C \supset D) \supset B \Rightarrow x(u.v.M, z.N) : E}\,L\supset\supset$$

These rules only construct well-formed terms; e.g. the notation $\Gamma, x : A \supset B, y : A$ in the conclusion of $L0\supset$ forces $x \neq y$ and x to be not already declared in Γ (hence not free in M).

These rules are the extensions with terms of the logical rules of **G4ip** in [TS00] (note a slight difference of the $L\supset\supset$ rule from that of [Dyc92]), with the variation that both in Ax and in $L0\supset$ the type A need not be atomic. In the rules $R\supset$, $L0\supset$ and $L\supset\supset$ the types $A \supset B$, $A \supset B$ and $(C \supset D) \supset B$ respectively are *principal*; in $L0\supset$ the type A is *auxiliary*. (This use of "auxiliary" is not to be confused with its use in Definition 1 to describe certain kinds of term.)

Note that in every instance of a rule in Definition 4 with conclusion $\Gamma \Rightarrow M : A$, each premiss $\Gamma' \Rightarrow N : B$ is such that $m(\Gamma) \cup A >_{\mathsf{mul}} m(\Gamma') \cup B$, where $\cup$ denotes the union of multi-sets. As a consequence, given Γ and A, there are finitely many derivations concluding, for some (normal) term M, the sequent $\Gamma \Rightarrow M : A$.

Definition 5 (Typing Rules for Auxiliary Terms).

$$\frac{\Gamma, y : C \supset D \Rightarrow M : E}{\Gamma, x : D \Rightarrow \mathtt{inv}(x, y.M) : E}\,Inv \qquad \frac{\Gamma \Rightarrow M : A \supset B}{\Gamma, x : A \Rightarrow \mathtt{of}(M, x) : B}\,Of$$

$$\frac{\Gamma, z : (C \supset D) \supset B \Rightarrow M : A}{\Gamma, x : C, y : D \supset B \Rightarrow \mathtt{dec}(x, y, z.M) : A}\,Dec \qquad \frac{\Gamma \Rightarrow M : A \quad x : A, \Gamma \Rightarrow N : B}{\Gamma \Rightarrow \mathtt{cut}(M, x.N) : B}\,Cut$$

These rules only construct well-formed terms; e.g. the notation $\Gamma, x : A$ in the conclusion of Inv forces x to be not declared in Γ and hence not free in M.

In the Cut-rule, we say that A is the *cut-type*. *Derivations* are built as usual from the rules (Definitions 4 and 5). A derivation is *normal* if it uses only the primitive rules, i.e. those of Definition 4. The *height* of a derivation is just its height as a tree; so a tree with one node has height 0.

We will occasionally find it necessary to rename free variables. The *renaming* by the variable y of all the free occurrences of x in M, written $\{y/x\}M$, is defined whenever y and x are distinct variables, M is a well-formed term and y is not free in M. This is an implicit substitution rather than explicit (i.e. a meta-notation rather than a term constructor). Renaming is sound with respect to typing, as shown by the first of the two following results of admissibility, in the standard sense [TS00].

Lemma 1. *The following rules are admissible both in the system of normal derivations and in the full system with auxiliary terms, with the proviso that $y \neq x$ in the* (Ren) *rule. (We use dashed lines and parenthesize the names of the rules to emphasise their admissibility in these systems.)*

$$\frac{\Gamma, x\!:\!B \Rightarrow M\!:\!A}{\Gamma, y\!:\!B \Rightarrow \{y/x\}M\!:\!A}\,(Ren) \qquad \frac{\Gamma \Rightarrow M\!:\!A}{\Gamma, y\!:\!B \Rightarrow M\!:\!A}\,(W)$$

Proof: Routine induction on the height of the derivation of the premiss. Some swapping of bound variable names may be necessary. Note that the notation $\Gamma, y\!:\!B$ forces y to be not declared in Γ and hence not free in M. $\square$

Remark 1. Note that for each proved sequent $\Gamma \Rightarrow M\!:\!A$ there is a unique derivation tree (up to renaming, in sub-derivations, of the variables bound in M), which can be reconstructed using the structure of the term M that *represents* the proof (hence the notion of *proof-term*).

3 Proof Transformations and Reduction Rules

The starting point of this section is the admissibility in the logical sequent calculus **G4ip** of the following inference rules (i.e. the logical counter-part of the typing rules for auxiliary terms given in Definition 5):

$$\frac{\Gamma, C\supset D \Rightarrow E}{\Gamma, D \Rightarrow E}\,Inv \qquad \frac{\Gamma \Rightarrow A\supset B}{\Gamma, A \Rightarrow B}\,\mathit{Of}$$

$$\frac{\Gamma, (C\supset D)\supset B \Rightarrow A}{\Gamma, C, D\supset B \Rightarrow A}\,Dec \qquad \frac{\Gamma \Rightarrow A \qquad A, \Gamma \Rightarrow B}{\Gamma \Rightarrow B}\,Cut$$

The admissibility of *Inv* and *Of* in **G4ip** can be proved, independently, by induction on the heights of the derivations of the premisses. For the admissibility of *Dec* and *Cut* we can use a simultaneous induction, the admissibility of one rule being recursively used for the admissibility of the other. The measure now uses the multi-set of types appearing in the unique premiss for *Dec* and in the second premiss for *Cut*. In other words, the induction can be done on $\{\!\{\Gamma, (C\supset D)\supset B, A\}\!\}$ for *Dec* and on $\{\!\{\Gamma, A, B\}\!\}$ for *Cut*.

We do not include here the detail of these proofs of admissibility, because the property turns out to be a consequence (Corollary 2) of our strong normalisation result for our calculus with terms.

Indeed, the admissibility property means, in our framework with terms, that a term M with auxiliary constructors `inv`, `of`, `dec` or `cut` can be transformed into another term M' with the same type in the same context that does not use these constructors.

This motivates the notion of *term-irrelevant admissibility* in a system with terms:

Definition 6. *A rule R is* term-irrelevantly admissible *in system S if, given an instance with conclusion $\Gamma \Rightarrow M\!:\!A$ and derivations in system S of its premiss(es), there exists a derivation in S of $\Gamma \Rightarrow M'\!:\!A$ for some term M'.*

Remark that this notion corresponds to the standard notion of admissibility when term annotations are erased.

Moreover, the inductive arguments of admissibility above can be seen as *weakly normalising* term reduction systems that specify how to eliminate the auxiliary constructors `inv`, `of`, `dec` and `cut`.

The reduction systems, given hereafter, must satisfy the following properties:

1. A term containing an auxiliary constructor is reducible by these systems.
2. They satisfy the Subject Reduction property, i.e. preservation of typing.
3. They satisfy some termination property.

Concerning point 3, the weak normalisation property of these systems suffices to prove the results of admissibility, and the proofs suggested above can be expressed as a terminating innermost strategy for these reduction systems. Nevertheless, we give in this paper reduction systems that are in fact *strongly normalising*. While this might be inferred for the orthogonal systems that we present in Section 6 (since weak innermost normalisation is equivalent to strong normalisation for orthogonal first-order systems [O'D77]), the result is certainly not so straightforward for the non-orthogonal ones. However, the measures for induction mentioned above can be taken as part of a *Multi-Set Path Ordering* [KL80,BN98] in order to conclude strong normalisation as well (see Section 4).

We give in Tables 1, 2 and 3 the reduction systems that eliminate the auxiliary constructors `of`, `inv` and `dec`. All these rules that we call system oid will be part of the different variants that we are going to introduce.

In order to reduce the cuts we now suggest a general system called cegs for cut-elimination in Tables 4 and 5 (variants are presented in Section 6). The whole system is called gs and contains the reduction rules in cegs (Tables 4 and 5) plus the ones in oid (Tables 1, 2 and 3).

Summing up :

Name of the System	Reduction Rules
oid	Tables 1, 2 and 3
cegs	Tables 4, 5
gs	$\mathsf{oid} \cup \mathsf{cegs}$

$\mathtt{of}(y, x)$	$\longrightarrow_{\mathsf{o1}}$	$y(x, z.z)$
$\mathtt{of}(\lambda y.M, x)$	$\longrightarrow_{\mathsf{o2}}$	$\{x/y\}M$
$\mathtt{of}(y(z, w.N), x)$	$\longrightarrow_{\mathsf{o3}}$	$y(z, w.\mathtt{of}(N, x))$
$\mathtt{of}(y(u.v.M, w.N), x)$	$\longrightarrow_{\mathsf{o4}}$	$y(u.v.M, w.\mathtt{of}(N, x))$

Table 1. Reduction Rules for $\mathtt{of}$-terms

$\mathtt{inv}(x, y.z)$	$\longrightarrow_{\mathsf{i1}}$	z
$\mathtt{inv}(x, y.y)$	$\longrightarrow_{\mathsf{i2}}$	$\lambda z.x$
$\mathtt{inv}(x, y.\lambda z.M)$	$\longrightarrow_{\mathsf{i3}}$	$\lambda z.\mathtt{inv}(x, y.M)$
$\mathtt{inv}(x, y.y(w, z.N))$	$\longrightarrow_{\mathsf{i4}}$	$\{x/z\}N$
$\mathtt{inv}(x, y.y(u.v.M, z.N))$	$\longrightarrow_{\mathsf{i5}}$	$\{x/z\}N$
$\mathtt{inv}(x, y.w(y, z.N))$	$\longrightarrow_{\mathsf{i6}}$	$w(u.v.x, z.\mathtt{inv}(x, y.N))$
$\mathtt{inv}(x, y.y'(w, z.N))$	$\longrightarrow_{\mathsf{i7}}$	$y'(w, z.\mathtt{inv}(x, y.N))$
$\mathtt{inv}(x, y.y'(u.v.M, z.N))$	$\longrightarrow_{\mathsf{i8}}$	$y'(u.v.\mathtt{inv}(x, y.M), z.\mathtt{inv}(x, y.N))$

Table 2. Reduction Rules for $\mathtt{inv}$-terms

$\mathtt{dec}(x, y, z.w)$	$\longrightarrow_{\mathsf{d1}}$	w
$\mathtt{dec}(x, y, z.z)$	$\longrightarrow_{\mathsf{d2}}$	$\lambda v.v(x, w.y(w, u.u))$
$\mathtt{dec}(x, y, z.\lambda w.M)$	$\longrightarrow_{\mathsf{d3}}$	$\lambda w.\mathtt{dec}(x, y, z.M)$
$\mathtt{dec}(x, y, z.w(u.v.M, w'.N))$	$\longrightarrow_{\mathsf{d4}}$	$w(u.v.\mathtt{dec}(x, y, z.M), w'.\mathtt{dec}(x, y, z.N))$
$\mathtt{dec}(x, y, z.w(y', z'.M))$	$\longrightarrow_{\mathsf{d5}}$	$w(y', z'.\mathtt{dec}(x, y, z.M))$
$\mathtt{dec}(x, y, z.z(y', z'.M))$	$\longrightarrow_{\mathsf{d6}}$	$y'(x, z''.y(z'', z'.\mathtt{inv}(z'', y'.M)))$
$\mathtt{dec}(x, y, z.x'(z, z'.M))$	$\longrightarrow_{\mathsf{d7}}$	$x(u.v.v(x, z''.y(z'', w.w)), z'.\mathtt{dec}(x, y, z.M))$
$\mathtt{dec}(x, y, z.z(u.v.M, z'.N))$	$\longrightarrow_{\mathsf{d8}}$	$\mathtt{cut}(\{x/u\}\{y/v\}M, y'.y(y', z'.N))$

Table 3. Reduction Rules for $\mathtt{dec}$-terms

$\mathsf{Kind_1}$		
$\mathtt{cut}(M, x.x)$	$\longrightarrow_{\mathsf{c1}}$	M
$\mathtt{cut}(M, x.y)$	$\longrightarrow_{\mathsf{c2}}$	y
$\mathtt{cut}(M, x.\lambda y.N)$	$\longrightarrow_{\mathsf{c3}}$	$\lambda y.\mathtt{cut}(M, x.N)$
$\mathtt{cut}(M, x.y(z, w.N))$	$\longrightarrow_{\mathsf{c4}}$	$y(z, w.\mathtt{cut}(\mathtt{inv}(w, y.M), x.N))$
$\mathtt{cut}(M, x.y(u.v.N', w.N))$	$\longrightarrow_{\mathsf{c5}}$	$y(u.v.P, w.\mathtt{cut}(\mathtt{inv}(w, y.M), x.N))$
		where $P = \mathtt{cut}(\mathtt{dec}(u, v, y.M), x.N')$
$\mathtt{cut}(\lambda z.M, x.y(x, w.N))$	$\longrightarrow_{\mathsf{c6}}$	$y(u.v.P, w.\mathtt{cut}(\mathtt{inv}(w, y.\lambda z.M), x.N))$
		where $P = \mathtt{cut}(u, z.\mathtt{dec}(u, v, y.M))$
$\mathtt{cut}(z, x.y(x, w.N))$	$\longrightarrow_{\mathsf{c7}}$	$y(z, w.\mathtt{cut}(z, x.N))$
$\mathsf{Kind_2}$		
$\mathtt{cut}(y(z, w.M), x.N)$	$\longrightarrow_{\mathsf{c8}}$	$y(z, w.\mathtt{cut}(M, x.\mathtt{inv}(w, y.N)))$
$\mathtt{cut}(y(u.v.M', w.M), x.N)$	$\longrightarrow_{\mathsf{c9}}$	$y(u.v.M', w.\mathtt{cut}(M, x.\mathtt{inv}(w, y.N)))$

Table 4. Cut Elimination Rules cegs (Kind_1 and Kind_2)

$\mathsf{Kind_3}$		
$\mathtt{cut}(M, x.x(z, w.N))$	$\longrightarrow_{A}$	$\mathtt{cut}(\mathtt{cut}(z, y.\mathtt{of}(M, y)), w.N)$
$\mathtt{cut}(M, x.x(u.v.N', w.N))$	$\longrightarrow_{B}$	$\mathtt{cut}(P, w.N)$
where		$P = \mathtt{cut}(\lambda u.\mathtt{cut}(\lambda z.\mathtt{inv}(z, y.\mathtt{of}(M, y)), v.N'), y.\mathtt{of}(M, y))$

Table 5. Cut Elimination Rules cegs (Kind_3)

As in most cut-elimination systems, the cut-reduction rules can be split into three kinds ($\mathsf{Kind}_1, \mathsf{Kind}_2, \mathsf{Kind}_3$), according to whether they push cuts to the right, to the left, or they break a cut into cuts on smaller types.

Here, owing to the particular inference rules of **G4ip** and the well-formedness constraints they impose on terms, the first two kinds must use the auxiliary constructs `inv` and `dec`, rather than just propagate the cuts.

For the third kind of cut-reduction rules, we usually expect both sub-proofs of the cut to introduce the cut-type (on the right and on the left, respectively). In particular, this requires the first argument of the cut-constructor to be a *value*, i.e. a variable or an abstraction, with a functional type, i.e. an implication $A \supset B$. However, just as *any* λ-term can be turned into a value by an η-expansion, here *any* term can be turned into a value by the use of the `of` constructor, with the following rule, which we also call η:

$$M \longrightarrow_\eta \lambda x.\mathtt{of}(M, x) \qquad \text{if } x \notin FV(M)$$

Note that in both cases this is only sound with respect to typing if the type of the original term is an implication.

Lemma 2. *All rules of system* gs *are such that well-formed terms reduce to well-formed terms.*

Proof. Routine.

4 A First-Order Syntax for Typed G4ip-Terms

Termination of the above rewrite systems on typed terms will be proved by the decrease of a measure associated to typing derivations. The latter are mapped to a first-order syntax with the following infinite signature:

$$\Sigma = \{\star/0, \mathsf{I}/1, \mathsf{K}/2, \mathsf{J}/1\} \cup \{\mathsf{D}^m/1, \mathsf{C}^m/2 \mid m \text{ is a multiset of types}\}$$

where the notation f/n is used to say that the symbol f has arity n, and the symbols have the following precedence relation:

$$\mathsf{C}^n \succ \mathsf{D}^n \succ \cdots \succ \cdots \succ \mathsf{C}^m \succ \mathsf{D}^m \succ \mathsf{J} \succ \mathsf{K} \succ \mathsf{I} \succ \star \qquad \text{if } n >_{\mathsf{mul}} m$$

The precedence relation on symbols provides a *Multi-set Path Ordering* $\gg_{mpo}$ (mpo) on first-order terms [KL80,BN98].

Remark 2.

1. The order on types (Def. 3) is well-founded, so $>_{\mathsf{mul}}$ is well-founded [DM79].
2. The order $>_{\mathsf{mul}}$ is well-founded, so $\succ$ is also well-founded.
3. The order $\succ$ is well-founded, so the Multi-Set Path Ordering $\gg_{mpo}$ is also well-founded.

Derivations are mapped to this first-order syntax. In particular, since each sequent $\Gamma \Rightarrow M\!:\!A$ has at most one derivation, we write $\overline{\Gamma \Rightarrow M\!:\!A}$ for such a translation, and even $\overline{M}$ when the context and type are clear from the text, as in the right-hand sides of the following definition.

$$
\begin{array}{ll}
\overline{\Gamma, x\!:\!A \Rightarrow x\!:\!A} & = \star \\
\overline{\Gamma \Rightarrow \lambda x.M\!:\!A\supset B} & = \mathsf{I}(\overline{M}) \\
\overline{\Gamma, x\!:\!A\supset B, y\!:\!A \Rightarrow x(y, z.M)\!:\!E} & = \mathsf{I}(\overline{M}) \\
\overline{\Gamma, x\!:\!(C\supset D)\supset B \Rightarrow x(u.v.M, z.N)\!:\!E} & = \mathsf{K}(\overline{M}, \overline{N}) \\
\overline{\Gamma, x\!:\!D \Rightarrow \mathtt{inv}(x, y.M)\!:\!E} & = \mathsf{J}(\overline{M}) \\
\overline{\Gamma, x\!:\!A \Rightarrow \mathtt{of}(M, x)\!:\!B} & = \mathsf{J}(\overline{M}) \\
\overline{\Gamma, x\!:\!C, y\!:\!D\supset B \Rightarrow \mathtt{dec}(x, y, z.M)\!:\!A} & = \mathsf{D}^k(\overline{\Gamma, z\!:\!(C\supset D)\supset B \Rightarrow M\!:\!A}) \\
 & \qquad\qquad \text{where } k = \{\!\{\Gamma, (C\supset D)\supset B, A\}\!\} \\
\overline{\Gamma \Rightarrow \mathtt{cut}(M, x.N)\!:\!B} & = \mathsf{C}^k(\overline{\Gamma \Rightarrow M\!:\!A}, \overline{x\!:\!A, \Gamma \Rightarrow N\!:\!B}) \\
 & \qquad\qquad\qquad\qquad \text{where } k = \{\!\{\Gamma, A, B\}\!\}
\end{array}
$$

Observe that $\overline{M} = \overline{\{x/y\}M}$ for any renaming of M.

5 Subject Reduction and Strong Normalisation

In this section we show two fundamental properties of system gs. The first one is subject reduction and it guarantees that types are preserved by the reduction system. The second one is strong normalisation and it guarantees that there is no infinite reduction sequence starting from a typed term. Strong normalisation is shown by a decreasing measure given by the Multi-Set Path Ordering of Section 4.

Theorem 1. *If $\Gamma \Rightarrow L\!:\!E$ and $L \longrightarrow_{\mathsf{gs}} L'$, then $\Gamma \Rightarrow L'\!:\!E$ and $\overline{L} \gg_{mpo} \overline{L'}$.*

Proof: By induction on the derivation of $\Gamma \Rightarrow L\!:\!E$. For brevity we show only the most important case of rule B, which reduces $\mathtt{cut}(M, x.x(u.v.N, z.N'))$ to $\mathtt{cut}(\mathtt{cut}(\lambda u.\mathtt{cut}(\lambda y'.\mathtt{inv}(y', y.\mathtt{of}(M, y)), v.N), y.\mathtt{of}(M, y)), z.N')$.

The derivation

$$
\dfrac{\dfrac{\dots}{\Gamma \Rightarrow M\!:\!(C\supset D)\supset B} \qquad \dfrac{\dfrac{\dots}{u\!:\!C, v\!:\!D\supset B, \Gamma \Rightarrow N\!:\!D} \qquad \dfrac{\dots}{z\!:\!B, \Gamma \Rightarrow N'\!:\!E}}{x\!:\!(C\supset D)\supset B, \Gamma \Rightarrow x(u.v.N, z.N')\!:\!E}\,L\supset\supset}{\Gamma \Rightarrow \mathtt{cut}(M, x.x(u.v.N, z.N'))\!:\!E}\,Cut
$$

rewrites to

$$
\dfrac{\dfrac{\dfrac{\mathcal{D}}{\Gamma \Rightarrow M'\!:\!C\supset D} \qquad \dfrac{\dfrac{\dots}{\Gamma \Rightarrow M\!:\!(C\supset D)\supset B}}{\Gamma, y\!:\!C\supset D \Rightarrow \mathtt{of}(M, y)\!:\!B}\,Of}{\Gamma \Rightarrow \mathtt{cut}(M', y.\mathtt{of}(M, y))\!:\!B}\,Cut \qquad \dfrac{\dots}{z\!:\!B, \Gamma \Rightarrow N'\!:\!E}}{\Gamma \Rightarrow \mathtt{cut}(\mathtt{cut}(M', y.\mathtt{of}(M, y)), z.N')\!:\!E}\,Cut
$$

where $M' = \lambda u.\mathtt{cut}(\lambda y'.\mathtt{inv}(y', y.\mathtt{of}(M, y)), v.N)$ and $\mathcal{D}$ is the following derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dots}{\Gamma \Rightarrow M:(C\supset D)\supset B}}{\Gamma, y:C\supset D \Rightarrow \mathtt{of}(M,y):B}\,\mathit{Of}}{\Gamma, u:C, y:C\supset D \Rightarrow \mathtt{of}(M,y):B}\,(W)}{\Gamma, u:C, y':D \Rightarrow \mathtt{inv}(y', y.\mathtt{of}(M,y)):B}\,\mathit{Inv}}{\Gamma, u:C \Rightarrow \lambda y'.\mathtt{inv}(y', y.\mathtt{of}(M,y)):D\supset B}\,R\supset \qquad \dfrac{\dots}{u:C, v:D\supset B, \Gamma \Rightarrow N:D}}{\dfrac{\Gamma, u:C \Rightarrow \mathtt{cut}(\lambda y'.\mathtt{inv}(y', y.\mathtt{of}(M,y)), v.N):D}{\Gamma \Rightarrow \lambda u.\mathtt{cut}(\lambda y'.\mathtt{inv}(y', y.\mathtt{of}(M,y)), v.N):C\supset D}\,R\supset}\,\mathit{Cut}$$

Let $k = \{\!\{(C\supset D)\supset B, \Gamma, E\}\!\}$ and $j = \{\!\{B, \Gamma, E\}\!\}$ and $i = \{\!\{\Gamma, C\supset D, B\}\!\}$ and $h = \{\!\{C, D\supset B, \Gamma, D\}\!\}$. Since $k >_{\mathsf{mul}} j, i, h$, we have $\mathsf{C}^k \succ \mathsf{I}, \mathsf{J}, \mathsf{C}^j, \mathsf{C}^i, \mathsf{C}^h$ and

$$\overline{L} = \mathsf{C}^k(\overline{M}, \mathsf{K}(\overline{N}, \overline{N'})) \gg_{mpo} \mathsf{C}^j(\mathsf{C}^i(\mathsf{I}(\mathsf{C}^h(\mathsf{I}(\mathsf{J}(\mathsf{J}(\overline{M}))), \overline{N})), \mathsf{J}(\overline{M})), \overline{N'}) = \overline{L'}$$

Full details can be found in [DKL06]. □

Corollary 1 (Strong Normalisation). *System* gs *is strongly normalising on typed terms.*

Proof: This is a consequence of Theorem 1 and Remark 2. □

Corollary 2. *Rules* $\mathit{Inv}, \mathit{Of}, \mathit{Dec}$, *and* Cut *are term-irrelevantly admissible in the system of Definition 4.*

Proof: Every term with an auxiliary constructor is reducible by system gs. □

6 Variants of reduction systems

We investigate in this section some variants of the cut-elimination system of Section 3.

We discuss in Section 6.1 the rules of Kind_3, noticing that the `of`-constructor is only introduced by the reductions of gs in order to include η-conversion in the system. We present two variations without η-conversion, called system rs and system ars, that no longer use the `of`-constructor.

Without η-conversion, the only critical pairs of those variations are between the rules of Kind_1 and those of Kind_2, so in Section 6.2, which only concerns rules of Kind_1 and Kind_2, we present two ways of removing those critical pairs, i.e. of making systems rs and ars orthogonal.

6.1 Avoiding the $\texttt{of}$-constructor

In this section we remove η-expansion from the reduction system so that the $\texttt{of}$-constructor is no more used by the cut elimination rules. We obtain two variants, depending on whether we want variables to behave like their η-expansions or we want the elimination of a cut with a variable to be simpler and closer to renaming.

The rules A and B of system gs (Table 5) introduce the $\texttt{of}$-constructor to model η-expansion, turning the first argument of the cut into an abstraction.

Theorem 2. *Rule A (resp. B) can be factorised into an η-expansion followed by rule C (resp. D) below:*

$$
\begin{array}{l}
\texttt{cut}(\lambda y.M, x.x(z, w.N)) \longrightarrow_C \texttt{cut}(\texttt{cut}(z, y.M), w.N) \\
\texttt{cut}(\lambda y.M, x.x(u.v.N', w.N)) \\
\qquad\qquad\qquad\qquad \longrightarrow_D \texttt{cut}(\texttt{cut}(\lambda u.\texttt{cut}(\lambda z.\texttt{inv}(z, y.M), v.N'), y.M), w.N)
\end{array}
$$

Proof:

$$
\begin{array}{lll}
\text{Rule } A: & & \texttt{cut}(M, x.x(z, w.N)) \\
& \longrightarrow_\eta & \texttt{cut}(\lambda y.\texttt{of}(M, y), x.x(z, w.N)) \\
& \longrightarrow_C & \texttt{cut}(\texttt{cut}(z, y.\texttt{of}(M, y)), w.N) \\
\text{Rule } B: & & \texttt{cut}(M, x.x(u.v.N', w.N)) \\
& \longrightarrow_\eta & \texttt{cut}(\lambda y.\texttt{of}(M, y), x.x(u.v.N', w.N)) \\
& \longrightarrow_D & \texttt{cut}(\texttt{cut}(\lambda u.\texttt{cut}(\lambda z.\texttt{inv}(z, y.\texttt{of}(M, y)), v.N'), y.\texttt{of}(M, y)), w.N)
\end{array}
$$

□

Note that the η-expansion of an abstraction reduces, by direct elimination of the $\texttt{of}$, to the abstraction itself:

$$\lambda y.M \longrightarrow_\eta \lambda x.\texttt{of}(\lambda y.M, x) \longrightarrow_{\mathsf{o2}} \lambda x.\{x/y\}M =_\alpha \lambda y.M \text{ with } x \notin FV(M)$$

This justifies the following theorem:

Theorem 3. *Rules C and D can be respectively derived from rules A and B using system* oid.

Proof: Similar to Theorem 2. □

Similarly, direct elimination of the $\texttt{of}$-constructor is allowed by rule $\mathsf{o1}$ in the case of a variable ($y \longrightarrow_\eta \lambda x.\texttt{of}(y, x) \longrightarrow_{\mathsf{o1}} \lambda x.y(x, z.z)$ with $x \notin FV(M)$), so this suggests that two rules E and F, treating the case of a variable, can also be derived from rules A and B:

Theorem 4. *The following rules E and F can be respectively derived from A and B using system* gs*:*

$$
\begin{array}{ll}
\texttt{cut}(y, x.x(z, w.N)) & \longrightarrow_E y(z, w'.\texttt{cut}(w', w.\texttt{inv}(w', y.N))) \\
\texttt{cut}(y, x.x(u.v.N', w.N)) & \longrightarrow_F y(u'.v'.\texttt{cut}(u', u.P), w'.\texttt{cut}(w', w.\texttt{inv}(w', y.N))) \\
& \quad \textit{where } P = \texttt{dec}(u', v', y.\texttt{cut}(\lambda y''.y(u.v.y'', z.z), v.N'))
\end{array}
$$

Proof: Similar to Theorem 2. □

Now, *by construction*, rules E and F make variables have the same functional behaviour as their η-expansion.

Note also that the new rules C, D, E and F (together with rules c8 and c9) can now replace any use of rules A and B, thus forming a system, called cers, that is still complete for cut-elimination and makes no use of the `of`-constructor. We show in Table 6 only the cut reduction rules of Kind_3, in which cegs and cers differ, the rules of Kind_1 and Kind_2 being the same. System cegs can thus be seen as system cers to which η-expansion has been integrated by the use of the auxiliary constructor `of`.

Kind_3	
$\mathtt{cut}(\lambda y.M, x.x(z, w.N))$	$\longrightarrow_C \mathtt{cut}(\mathtt{cut}(z, y.M), w.N)$
$\mathtt{cut}(\lambda y.M, x.x(u.v.N', w.N))$	$\longrightarrow_D \mathtt{cut}(\mathtt{cut}(\lambda u.\mathtt{cut}(\lambda z.\mathtt{inv}(z, y.M), v.N'), y.M), w.N)$
$\mathtt{cut}(y, x.x(z, w.N))$	$\longrightarrow_E y(z, w'.\mathtt{cut}(w', w.\mathtt{inv}(w', y.N)))$
$\mathtt{cut}(y, x.x(u.v.N', w.N))$	$\longrightarrow_F y(u'.v'.\mathtt{cut}(u', u.P), w'.\mathtt{cut}(w', w.\mathtt{inv}(w', y.N)))$
	where $P = \mathtt{dec}(u', v', y.\mathtt{cut}(\lambda y''.y(u.v.y'', z.z), v.N'))$

Table 6. Cut Elimination Rules in System cers (Kind_3)

The behaviour of functionals is interesting in **G4ip**, because it is a depth-bounded calculus: for instance, among all Church's numerals only 0 and 1 can be represented in **G4ip**. So when reducing the term that represents (using cuts) "$1 + 1$", we should expect some semantical anomaly in the reductions (which is quite similar to the one reported by Vestergaard in [Ves99]). Such an anomaly is to be found in rules B and D, and for abstractions we have no alternative choice. However in system rs we have made the choice of making variables have the same functional behaviour as their η-expansions, hence rule F inherits the anomaly. But instead we might rather follow the intuition that cutting a variable with a another variable is almost renaming, and replace rule F with a new rule G, thus forming system cears presented in Table 7 (again we only show rules of Kind_3, but rules of Kind_1 and Kind_2 are the same as in cegs or cers). This new rule is simpler and more natural than rule F; however the reducts are semantically different and thus the choice of rule G breaks the property that a variable and its η-expansion have the same behaviour.

Since all the rules of system rs are derived from system gs, it is clear that the former inherits from the latter the Subject Reduction property as well as the Strong Normalisation of typed terms. However, for system ars, those properties are not inherited, even if it is easy to check that rule G satisfies the Subject Reduction property and decreases with respect to the multi-set path ordering in Section 4.

Kind_3	
$\mathtt{cut}(\lambda y.M, x.x(z, w.N))$	$\longrightarrow_C \mathtt{cut}(\mathtt{cut}(z, y.M), w.N)$
$\mathtt{cut}(\lambda y.M, x.x(u.v.N', w.N))$	$\longrightarrow_D \mathtt{cut}(\mathtt{cut}(\lambda u.\mathtt{cut}(\lambda z.\mathtt{inv}(z, y.M), v.N'), y.M), w.N)$
$\mathtt{cut}(y, x.x(z, w.N))$	$\longrightarrow_E y(z, w'.\mathtt{cut}(w', w.\mathtt{inv}(w', y.N)))$
$\mathtt{cut}(y, x.x(u.v.N', w.N))$	$\longrightarrow_G y(u'.v'.\mathtt{cut}(u', u.P'), w'.\mathtt{cut}(w', w.\mathtt{inv}(w', y.N)))$
	where $P' = \mathtt{cut}(v', v.\mathtt{dec}(u', v', y.N'))$

Table 7. Cut Elimination Rules in System cears (Kind_3)

The systems presented so far in this paper can be summarised in the following table:

$\mathtt{of}$, $\mathtt{inv}$ and $\mathtt{dec}$	$\mathtt{cut}$ =	(Kind_1 + Kind_2)	+ Kind_3	Whole system
oid	cegs =	Table 4	+ Table 5	gs
oid	cers =	Table 4	+ Table 6	rs
oid	cears =	Table 4	+ Table 7	ars

6.2 Orthogonal systems

In this section we suggest two ways of restricting the rules of Kind_1 and Kind_2 to make systems rs and ars orthogonal, and hence confluent.

In the restricted systems gs and ars there are overlaps between the right and left propagation sub-systems, i.e. there is a critical pair between any rule in $\{\mathsf{c1}, \mathsf{c2}, \mathsf{c3}, \mathsf{c4}, \mathsf{c5}\}$ and any rule in $\{\mathsf{c8}, \mathsf{c9}\}$. This is shown in Table 8, where column headers represent the different cases concerning the first premiss of the cut, while row headers represent the different cases for the second one (marking inside parentheses the status of the cut-type).

	Axiom	$R\supset$	$L0\supset$	$L\supset\supset$
Axiom (Principal)	c1	c1	c1, c8	c1, c9
Axiom (Non-Principal)	c2	c2	c2, c8	c2, c9
$R\supset$	c3	c3	c3, c8	c3, c9
$L0\supset$ (Non-Principal, Non-Auxiliary)	c4	c4	c4, c8	c4, c9
$L\supset\supset$ (Non-Principal)	c5	c5	c5, c8	c5, c9
$L0\supset$ (Non-Principal, Auxiliary)	c7	c6	c8	c9
$L0\supset$ (Principal)	E	C	c8	c9
$L\supset\supset$ (Principal)	F (rs) or G (ars)	D	c8	c9

Table 8. Overlaps of reduction rules

The overlaps pointed out in Table 8 are well-known in sequent calculus, and correspond to the choice of whether to push a cut into the proof of its left premiss or into the proof of its right premiss. The former corresponds to a *call-by-value* strategy and the latter corresponds to a *call-by-name* strategy.

Since the overlaps only concerns cut-reduction rules of Kind_1 and Kind_2, we discuss in the following possible ways to make them non-overlapping.

Call-by-name One way to make the system orthogonal is to give preference to rules c1-c2-c3-c4-c5 over rules c8-c9, thus restricted to the case when N is an x-covalue Q, i.e. is of the form $x(y, w.N)$ or $x(u.v.M, w.N)$.

Note that in order to reduce a term like $\mathtt{cut}(M, x.y(x, w.N))$, there is no choice other than left-propagation (rules c8 and c9) until a similar redex is found in which M is a value, and then only rules c6 or c7 can be applied.

Call-by-value Alternatively, preference might be given to rules c8 and c9, which we can formalise as restricting rules c1-c2-c3-c4-c5 to the case when M is a value V (variable or abstraction).

The choice of call-by-value is more natural than that of call-by-name because the two rules of right-propagation c6 and c7 only apply to cuts whose first argument is a value. This suggests that **G4ip** has an inherent *call-by-value* flavour, echoing the idea that it is somehow based on the call-by-value sequent calculus **LJQ**. Indeed, completeness of **LJQ** gives a short proof of the completeness of **G4ip** [DL06].

We finish this section by stating the following property of cbn and cbv.

Theorem 5. *Reduction systems* cbn *and* cbv *are confluent.*

Proof: Systems cbn and cbv can be seen as particular *orthogonal CRS*, so they enjoy confluence (see [vOvR94] for details). $\square$

7 Conclusion

This paper defines various term calculi for the depth-bounded intuitionistic sequent calculus of Hudelmaier. Using standard techniques of rewriting, we prove subject-reduction and strong normalisation for all of them, so *Cut*-admissibility turns out to be a corollary. The cbn and cbv systems presented in this paper are also orthogonal, which guarantees confluence (and uniqueness of normal forms).

Some relations between **G4ip** and other calculi for intuitionistic logic are studied in [DL06]. Also, from our term calculi for **G4ip**, which use explicit operators, we could extract term calculi with *implicit* operators (as in λ-calculus). This would bring our calculus closer to that of Matthes [Mat02], and with a strong normalising cut-elimination procedure. As mentioned in the introduction, defining a denotational semantics for our calculi as well as investigating the connexions with the simply-typed λ-calculus would reveal more properties of the proofs in **G4ip**.

References

[BN98] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[DKL06] R. Dyckhoff, D. Kesner, and S. Lengrand. Strong cut-elimination systems for Hudelmaier's depth-bounded sequent calculus for implicational logic, 2006. Full version. Available at http://www.pps.jussieu.fr/~lengrand/Work/Papers.html.

[DL06] R. Dyckhoff and S. Lengrand. **LJQ**, a strongly focused calculus for intuitionistic logic. In A. Beckmann, U. Berger, B. Loewe, and J. V. Tucker, editors, *Proc. of the 2nd Conf. on Computability in Europe (CiE'06)*, volume 3988 of *LNCS*. Springer-Verlag, July 2006.

[DM79] N. Dershowitz and Z. Manna. Proving termination with multiset orderings. *Communications of the ACM*, 22(8):465–476, 1979.

[DN00] R. Dyckhoff and S. Negri. Admissibility of structural rules for contraction-free systems of intuitionistic logic. *The Journal of Symbolic Logic*, 65(4):1499–1518, 2000.

[Dyc92] R. Dyckhoff. Contraction-free sequent calculi for intuitionistic logic. *The Journal of Symbolic Logic*, 57(3):795–807, 1992.

[Hud89] J. Hudelmaier. *Bounds for Cut Elimination in Intuitionistic Logic*. PhD thesis, Universität Tübingen, 1989.

[Hud92] J. Hudelmaier. Bounds on cut-elimination in intuitionistic propositional logic. *Archive for Mathematical Logic*, 31:331–354, 1992.

[KL80] S. Kamin and J.-J. Lévy. Attempts for generalizing the recursive path orderings. Handwritten paper, University of Illinois, 1980.

[LSS91] P. Lincoln, A. Scedrov, and N. Shankar. Linearizing intuitionistic implication. In *Proc. of the Sixth Annual IEEE Symposium on Logic in Computer Science*, pages 51–62, Amsterdam, The Netherlands, 1991.

[Mat02] R. Matthes. Contraction-aware λ-calculus, 2002. Seminar at Oberwolfach.

[O'D77] M. J. O'Donnell. *Computing in Systems Described by Equations*, volume 58 of *LNCS*. Springer-Verlag, 1977.

[ORK05] J. Otten, T. Raths, and C. Kreitz. The ILTP Library: Benchmarking automated theorem provers for intuitionistic logic. In B. Beckert, editor, *International Conference TABLEAUX-2005*, volume 3702 of *LNAI*, pages 333–337. Springer Verlag, 2005.

[Pit92] A. M. Pitts. On an interpretation of second order quantification in first-order intuitionistic propositional logic. *Journal of Symbolic Logic*, 57:33–52, 1992.

[Pit03] A. M. Pitts. Nominal logic, a first order theory of names and binding. *Information and Computation*, 186:165–193, 2003.

[TS00] A. S. Troelstra and H. Schwichtenberg. *Basic Proof Theory*. Cambridge University Press, 2000.

[Ves99] R. Vestergaard. Revisiting Kreisel: A computational anomaly in the Troelstra-Schwichtenberg **g3i** system, March 1999. Available at http://www.cee.hw.ac.uk/~jrvest/.

[Vor70] N. N. Vorob'ev. A new algorithm for derivability in the constructive propositional calculus. *American Mathematical Society Translations*, 94(2):37–71, 1970.

[vOvR94] V. van Oostrom and F. van Raamsdonk. Weak orthogonality implies confluence: the higher-order case. In A. Nerode and Y. Matiyasevich, editors, *Proc. of the 3rd Int. Symp. on Logical Foundations of Computer Science*, volume 813 of *LNCS*, pages 379–392. Springer-Verlag, July 1994.