Strong Normalisation of Cut-Elimination That Simulates β-Reduction

Kentaro Kikuchi[1] and Stéphane Lengrand[2]

[1] RIEC, Tohoku University, Japan
[2] CNRS, Laboratoire d'Informatique de l'Ecole Polytechnique, France

Abstract. This paper is concerned with strong normalisation of cut-elimination for a standard intuitionistic sequent calculus. The cut-elimination procedure is based on a rewrite system for proof-terms with cut-permutation rules allowing the simulation of β-reduction. Strong normalisation of the typed terms is inferred from that of the simply-typed λ-calculus, using the notions of *safe* and *minimal* reductions as well as a simulation in Nederpelt-Klop's λI-calculus. It is also shown that the type-free terms enjoy the preservation of strong normalisation (PSN) property with respect to β-reduction in an isomorphic image of the type-free λ-calculus.

1 Introduction

It is now established that cut-elimination procedures in sequent calculus have a computational meaning (see e.g. [12,7,32,26]), in the same sense as that of proof transformations in natural deduction. The paradigm of the Curry-Howard correspondence is then illustrated not only by (intuitionistic implicational) natural deduction and the simply-typed λ-calculus [13], but also by a typed higher-order calculus corresponding to the (intuitionistic implicational) sequent calculus.

In [16], the first author identified through a Prawitz-style translation a subset of proofs in a standard sequent calculus that correspond to simply-typed λ-terms, and defined a reduction relation on those proofs that precisely corresponds to β-reduction of the simply-typed λ-calculus. The reduction relation was shown to be simulated by a cut-elimination procedure, so the system of proof-terms for the sequent calculus is a conservative extension of the λ-calculus in both term-structure and reduction. Since the correspondence holds also for the type-free case, the rewrite system in [16] can simulate β-reduction of the type-free λ-calculus, which means that it is strong enough to represent all computable functions. It was also shown in [17] that a restriction of the rewrite system in [16], which is still strong enough to simulate β-reduction, is confluent.

The present paper presents the first proof of strong normalisation of the cut-elimination procedure in [17]. Since the cut-elimination procedure can simulate β-reduction of the simply-typed λ-calculus, its strong normalisation is at least as hard as that of the latter. In fact, the proof we develop in this paper relies on strong normalisation of the simply-typed λ-calculus. However, a naive simulation of the cut-elimination procedure by β-reduction fails, so we refine

R. Amadio (Ed.): FOSSACS 2008, LNCS 4962, pp. 380–394, 2008.

the approach by using two techniques formalised in [21,22] as the *Safeness & Minimality technique* and the *simulation in the λI-calculus* of [19] through a non-deterministic encoding. A by-product of our method is a proof of strong normalisation of the type-free terms that encode strongly normalising type-free λ-terms through the Prawitz-style translation. This is known as the property of *Preservation of Strong Normalisation* (PSN) in the field of calculi with explicit substitutions.

Strong normalisation of cut-elimination has been studied by a number of authors. Here we mention some of them that treat cut-elimination procedures consisting of (Gentzen-style) local proof transformations in a standard sequent calculus. The first is Dragalin's cut-elimination procedure and simple proof of its strong normalisation, which can be found in Chapter 7 of [28]. However, the cut-elimination procedure does not allow any permutation of cuts, so cannot simulate β-reduction. A cut-elimination procedure that simulates β-reduction can be found in [31] (for the classical sequent calculus), with a proof of strong normalisation. However, the cut-permutation rules involve extra kinds of cuts that are allowed to pass over usual cuts; therefore it is not clear how the proof of strong normalisation could be adapted to our case, which leaves the simple syntax of the calculus untouched. Recently, [23] introduced another cut-elimination procedure and proved its strong normalisation. The cut-elimination procedure is a modification of the one in [16], but differs from the one in the present paper. The proof technique for strong normalisation in [23] does not work for our system, and our proof in this paper solves a related problem that was explicitly given in Section 6 of [23]. Finally, [29] presents a proof of strong normalisation for a cut-elimination system that is not intended (and is unlikely) to simulate β-reduction. However, their technique is also inspired by Nederpelt and Klop's works on λI and how it compares to ours, different though the cut-elimination systems are, remains to be investigated.

The structure of the paper is as follows. In Section 2 we introduce a term assignment system for a standard sequent calculus and a rewrite system for a cut-elimination procedure in the sequent calculus. In Section 3 we explain our proof techniques and apply them to showing strong normalisation for the typed terms and the PSN property for the type-free terms. In Section 4 we discuss related work and conclude in Section 5.

To save space we omit some of the details in proofs, but a longer paper [18] is available at `http://www.lix.polytechnique.fr/~lengrand/Work/`.

2 A Rewrite System for Cut-Elimination

2.1 Term Assignment for Sequent Calculus

We start by giving a proof-term assignment system for a standard intuitionistic sequent calculus, following [16]. The syntax of proof-terms can be found in various textbooks (e.g. [30,28]) and papers (e.g. [10]) with notational variants. Here we call the proof-terms λG3-*terms*. Our cut-elimination procedure is represented as reduction rules for typed λG3-terms.

First, the set of raw $\lambda\mathsf{G3}$-terms is defined by the grammar:

$$M ::= x \mid \lambda x.M \mid \langle xM/x\rangle M \mid [M/x]M$$

where x ranges over a denumerable set of variables. $\langle \_\,\_/\_\rangle\_$ and $[\_/\_]\_$ are term constructors similar to explicit substitutions ($[\_/\_]\_$ is called the cut-constructor). We use letters x, y, z, w for variables and M, N, P, Q for $\lambda\mathsf{G3}$-terms. To denote that M is a strict subterm of N, we write $M \sqsubset N$ or $N \sqsupset M$. The notions of free and bound variables are defined as usual, with an additional clause that the variable x in $\langle yM/x\rangle N$ or $[M/x]N$ binds the free occurrences of x in N. The set of free variables of a $\lambda\mathsf{G3}$-term M is denoted by $\mathsf{FV}(M)$. We often use the notation $\langle \underline{x}M/y\rangle N$ to denote $\langle xM/y\rangle N$ if $x \notin \mathsf{FV}(M) \cup \mathsf{FV}(N)$. The symbol $\equiv$ denotes syntactical equality modulo α-conversion; so for example $\langle zP/x\rangle\langle \underline{x}M/y\rangle N \equiv \langle zP/w\rangle\langle \underline{w}M/y\rangle N$.

The proof-term assignment system for a standard intuitionistic sequent calculus is given in Figure 1. We define a typing context, ranged over by Γ, as a finite set of pairs $\{x_1 : A_1, \ldots, x_n : A_n\}$ where the variables are pairwise distinct. $\Gamma, x : A$ denotes the union $\Gamma \cup \{x : A\}$, and $x \notin \Gamma$ means that x does not appear in Γ. For precise representation of proofs by terms, we should specify formulas on binders, but we will omit them for brevity. If $x \notin \mathsf{FV}(M) \cup \mathsf{FV}(N)$ in the $\lambda\mathsf{G3}$-term $\langle xM/y\rangle N$, we assume $x \notin \Gamma$ in the rule $\mathsf{L}\supset$, which means the formula $A \supset B$ is introduced without implicit contraction.

$$\mathsf{Ax}\ \frac{}{\Gamma, x : A \vdash x : A} \qquad \mathsf{L}\supset\ \frac{\Gamma \vdash M : A \quad \Gamma, y : B \vdash N : C}{\Gamma, x : A \supset B \vdash \langle xM/y\rangle N : C}\ y \notin \Gamma$$

$$\mathsf{R}\supset\ \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \supset B}\ x \notin \Gamma \qquad \mathsf{Cut}\ \frac{\Gamma \vdash M : A \quad \Gamma, x : A \vdash N : B}{\Gamma \vdash [M/x]N : B}\ x \notin \Gamma$$

Fig. 1. Proof-term assignment for sequent calculus

We write $\Gamma \vdash_{\lambda\mathsf{G3}} M : A$ if the sequent is derivable with the inference rules of Figure 1. We also write $\Gamma \vdash_{\lambda} t : A$ if it is derivable with the standard inference rules of the simply-typed λ-calculus.

In order to understand the semantics of $\lambda\mathsf{G3}$, we can re-express Gentzen's translation of proofs in sequent calculus to those in natural deduction using $\lambda\mathsf{G3}$-terms and λ-terms ($\{-/\_\}\_$ means usual implicit substitution).

$$\begin{array}{lcl}
\mathcal{G}^1(x) & := & x \\
\mathcal{G}^1(\lambda x.M) & := & \lambda x.\mathcal{G}^1(M) \\
\mathcal{G}^1(\langle xM/y\rangle N) & := & \left\{{}^{x\ \mathcal{G}^1(M)}/_{y}\right\}\mathcal{G}^1(N) \\
\mathcal{G}^1([M/x]N) & := & \left\{{}^{\mathcal{G}^1(M)}/_{x}\right\}\mathcal{G}^1(N)
\end{array}$$

Notice that terms of $\lambda\mathsf{G3}^{\mathsf{cf}}$ (i.e. $\lambda\mathsf{G3}$-terms without the cut-constructor) are always mapped to λ-terms in normal form.

We can also give a backward translation from natural deduction to sequent calculus:

$$\begin{array}{lcl}\mathcal{G}^2(x) & := & x\\ \mathcal{G}^2(\lambda x.t) & := & \lambda x.\mathcal{G}^2(t)\\ \mathcal{G}^2(t\ s) & := & [\mathcal{G}^2(t)/x]\langle \underline{x}\ \mathcal{G}^2(s)/y\rangle y\end{array}$$

In the above translation, normal forms of λ-calculus are not necessarily mapped to $\lambda\mathsf{G3}^{\mathsf{cf}}$-terms. This is fixed by a Prawitz-style encoding:

$$\begin{array}{lcl}\mathcal{P}r(x) & := & x\\ \mathcal{P}r(\lambda x.t) & := & \lambda x.\mathcal{P}r(t)\\ \mathcal{P}r(t\ s) & := & \mathcal{P}r_{x.x}(t\ s)\\ \hline \mathcal{P}r_{x.M}(y\ s) & := & \langle y\mathcal{P}r(s)/x\rangle M\\ \mathcal{P}r_{x.M}((\lambda y.t)\ s) & := & [\lambda y.\mathcal{P}r(t)/z]\langle \underline{z}\mathcal{P}r(s)/x\rangle M\\ \mathcal{P}r_{x.M}(t_1\ t_2\ s) & := & \mathcal{P}r_{z.\langle \underline{z}\mathcal{P}r(s)/x\rangle M}(t_1\ t_2)\end{array}$$

Theorem 1 (Preservation of typing)

- *If* $\Gamma \vdash_{\lambda G3} M{:}A$ *then* $\Gamma \vdash_{\lambda} \mathcal{G}^1(M){:}A$.
- *If* $\Gamma \vdash_{\lambda} t{:}A$ *then* $\Gamma \vdash_{\lambda G3} \mathcal{G}^2(t){:}A$.
- *If* $\Gamma \vdash_{\lambda} t\ s{:}A$ *and* $\Gamma, x{:}A \vdash_{\lambda G3} M{:}B$ *then* $\Gamma \vdash_{\lambda G3} \mathcal{P}r_{x.M}(t\ s){:}B$.
- *If* $\Gamma \vdash_{\lambda} t{:}A$ *then* $\Gamma \vdash_{\lambda G3} \mathcal{P}r(t){:}A$.

The following theorems can be found in the literature, some of them in [10].

Theorem 2 (Properties of the encodings)

- $\mathcal{G}^1$ *is surjective, its restriction* $\mathcal{G}^1{}_{|\lambda G3^{\mathsf{cf}}}$ *to cut-free terms is surjective on normal* λ*-terms, and neither is injective.*
- $\mathcal{G}^2$ *and* $\mathcal{P}r$ *are both injective but not surjective on* $\lambda G3$.
- $\mathcal{G}^1 \circ \mathcal{G}^2 = \mathsf{Id}_{\lambda}$ *and* $\mathcal{G}^1 \circ \mathcal{P}r = \mathsf{Id}_{\lambda}$.
- *Neither* $\mathcal{G}^2 \circ \mathcal{G}^1 \neq \mathsf{Id}_{\lambda G3}$ *nor* $\mathcal{P}r \circ \mathcal{G}^1 \neq \mathsf{Id}_{\lambda G3}$.
- $\mathcal{G}^2 \circ \mathcal{G}^1$ *does not leave* $\lambda G3^{\mathsf{cf}}$ *stable*[1] *but* $\mathcal{P}r \circ \mathcal{G}^1$ *does.*

2.2 The Cut-Elimination Procedure

Our cut-elimination procedure is based on a rewrite system for $\lambda\mathsf{G3}$-terms. The system is the same as the one in [17], which is a confluent restriction of the system in [16]. (Although confluence is not used in this paper, the system in [16] so far seems to resist the present technique.)

[1] (i.e. if M is cut-free, $\mathcal{G}^2(\mathcal{G}^1(M))$ might not be).

Figure 2 shows the reduction rules of the rewrite system. Each of these reduction rules corresponds to a local cut-elimination step (cf. [18]). The reduction rules (1) through (5) correspond to cut-elimination steps that permute a cut upwards through its right subproof. The rules (6) and (7′) correspond to steps permuting a cut upwards through its left subproof. The rule (B) corresponds to the key-case which breaks a cut on an implication into two cuts on its subformulas. The rules (Perm_1) and (Perm_2) permute two cuts with some restrictions. In (Perm_1), the left rule over the lower cut is another cut, and the right rules over both cuts must be $\mathsf{L}\supset$ that introduces the cut-formula without implicit contraction. In (Perm_2), the right rule over the lower cut is another cut, which must construct a proof corresponding to a redex of the rule (B).

(1)	$[M/x]y \to y \quad (y \not\equiv x)$
(2)	$[M/x]x \to M$
(3)	$[N/x](\lambda y.M) \to \lambda y.[N/x]M$
(4)	$[P/z]\langle xM/y\rangle N \to \langle x([P/z]M)/y\rangle [P/z]N \quad (x \not\equiv z)$
(5)	$[P/x]\langle xM/y\rangle N \to [P/x]\langle \underline{x}([P/x]M)/y\rangle [P/x]N \quad$ if $x \in \mathsf{FV}(M) \cup \mathsf{FV}(N)$
(6)	$[z/x]\langle \underline{x}M/y\rangle N \to \langle zM/y\rangle N$
(7′)	$[\langle xM/y\rangle N/z]\langle \underline{z}M'/w\rangle N' \to \langle xM/y\rangle [N/z]\langle \underline{z}M'/w\rangle N'$
(B)	$[\lambda z.P/x]\langle \underline{x}M/y\rangle N \to [[M/z]P/y]N$
(Perm_1)	$[[P/x]\langle \underline{x}M/y\rangle N/z]\langle \underline{z}M'/w\rangle N' \to [P/x][\langle \underline{x}M/y\rangle N/z]\langle \underline{z}M'/w\rangle N'$
(Perm_2)	$[Q/w][\lambda z.P/x]\langle \underline{x}M/y\rangle N \to [[Q/w](\lambda z.P)/x][Q/w]\langle \underline{x}M/y\rangle N$

Fig. 2. Rewrite system for cut-elimination

The reduction relation $\longrightarrow_{\mathsf{cut}}$ is defined by the contextual closures of the reduction rules in Figure 2. We use $\longrightarrow^{+}_{\mathsf{cut}}$ for its transitive closure, and $\longrightarrow^{*}_{\mathsf{cut}}$ for its reflexive transitive closure. The set of λG3-terms that are strongly normalising with respect to $\longrightarrow_{\mathsf{cut}}$ is denoted by $\mathsf{SN}^{\mathsf{cut}}$. These kinds of notations are also used for the notions of other reductions in this paper.

The rewrite system without the rule (B) is called x. It was shown in [17] that the system x is strongly normalising and confluent.

The original rewrite system in [16] has instead of (7′) the rule (7) which is obtained by replacing $\langle \underline{z}M'/w\rangle N'$ in (7′) by a general term P. However then the system becomes non-confluent (e.g. the critical pair $w \leftarrow [\langle xM/y\rangle N/z]w \to \langle xM/y\rangle [N/z]w$ is not joinable). We study in this paper the system with (7′), which was shown to be confluent in [17] and which is still strong enough to simulate β-reduction.

Theorem 3 (Simulation of β-reduction)
$\longrightarrow_{cut}$ strongly simulates $\longrightarrow_{\beta}$ through the translation $\mathcal{P}r$, i.e. if $M \longrightarrow_{\beta} M'$ then $\mathcal{P}r(M) \longrightarrow^{+}_{cut} \mathcal{P}r(M')$.

Proof. This is a minor variant of the proof in [16]. The proof is by induction on the derivation of the reduction step, using various lemmas. □

3 Strong Normalisation

In this section we prove strong normalisation of $\longrightarrow_{\mathsf{cut}}$ on (typed) λG3-terms. Since this reduction relation can simulate β-reduction in λ-calculus, its strong normalisation is at least as hard as that of the simply-typed λ-calculus. In fact, our proof relies on the latter.

A by-product of our method is a proof of strong normalisation of those λG3-terms that encode strongly normalising type-free λ-terms through the Prawitz-style translation. This is known as the property of *Preservation of Strong Normalisation* (PSN) [3]. In other words, the reduction relation of λG3 is big enough to simulate β-reduction through the Prawitz-style translation, but small enough to be strongly normalising.

The basic idea in proving that a term M of λG3 is SN is to simulate the reduction steps from M by β-reduction steps from a strongly normalising λ-term $\mathcal{G}^1(M)$. Indeed, this would be relevant for PSN since $\mathcal{G}^1(\mathcal{P}r(t)) = t$, as well as for the strong normalisation of a typed λG3-term M, since $\mathcal{G}^1(M)$ is a simply-typed λ-term. The idea of simulating cut-elimination by β-reductions has been investigated in [35,25].

Unfortunately, Gentzen's encoding into λ-calculus, which allows the simulation, needs to interpret cut-constructors (and constructors for $\mathsf{L}\supset$) as implicit substitutions such as $\{{}^{u}\!/_{x}\}t$. Should x not be free in t, all reduction steps taking place within the term of which u is the encoding would not induce any β-reduction in $\{{}^{u}\!/_{x}\}t$. Therefore, the reduction relation that is only weakly simulated, i.e. the one consisting of all the reductions that are not necessarily simulated by at least one β-reduction, is too big to be proved terminating (in fact, it is not).

In order to overcome the aforementioned problem, we combine two techniques formalised in [21,22] as the *Safeness & Minimality technique* and the *simulation in the λI-calculus* of [19] through a non-deterministic encoding.

3.1 Safeness and Minimality

We first define safe and minimal reductions for the rewrite system of $\longrightarrow_{\mathsf{cut}}$ on (some class of) λG3-terms.

The intuitive idea is that a reduction step is *minimal* if all the (strict) subterms of the redex are in $\mathsf{SN}^{\mathsf{cut}}$. Theorem 4(1) says that in order to prove that $\longrightarrow_{\mathsf{cut}}$ is terminating, we can restrict our attention to minimal reductions only, without loss of generality.[2] Similarly, a reduction step is *safe* if the redex itself is in $\mathsf{SN}^{\mathsf{cut}}$, which is a stronger requirement than minimality. Theorem 4(2) says that

[2] Note that a perpetual strategy, in the sense of [33], can be defined so that only minimal reductions are performed. Also, the technique seems close to that of dependency pairs (see e.g. [1]) and formal connections should be studied.

safe reductions always terminate. Those ideas are made precise in the following definition:

Definition 1 (Safe and minimal reduction). *Given a subsystem h of our cut-elimination system, we define the following rules:*

$$\mathsf{minh} \quad M \longrightarrow N \quad \text{if } M \longrightarrow_{\mathsf{h}} N \text{ and for all } P \sqsubset M,\ P \in \mathsf{SN}^{\mathsf{cut}}$$
$$\mathsf{safeh} \quad M \longrightarrow N \quad \text{if } M \longrightarrow_{\mathsf{h}} N \text{ and } M \in \mathsf{SN}^{\mathsf{cut}}$$

and denote their contextual closures by $\longrightarrow_{\mathsf{minh}}$ and $\longrightarrow_{\mathsf{safeh}}$ respectively.

We say that a reduction step $M \longrightarrow_{\mathsf{h}} N$ is safe (resp. minimal) if $M \longrightarrow_{\mathsf{safeh}} N$ (resp. $M \longrightarrow_{\mathsf{minh}} N$) and that it is unsafe if not.[3]

Remark 1. We shall constantly use the following facts:

1. $\longrightarrow_{\mathsf{min(safeh)}} = \longrightarrow_{\mathsf{safe(minh)}} = \longrightarrow_{\mathsf{safeh}}$
2. $\longrightarrow_{\mathsf{safe(h,h')}} = \longrightarrow_{\mathsf{safeh,safeh'}}$
3. $\longrightarrow_{\mathsf{min(h,h')}} = \longrightarrow_{\mathsf{minh,minh'}}$

We have the following theorems (proofs can be found in [21,22]):

Theorem 4. *1. $\mathsf{SN}^{\mathsf{mincut}} = \mathsf{SN}^{\mathsf{cut}}$.*
2. For every λG3-term M, $M \in \mathsf{SN}^{\mathsf{safecut}}$.

In other words, safe reductions terminate, and in order to prove that a term is strongly normalising, it suffices to prove that it is strongly normalising for minimal reductions only.

This leads directly to the following corollary:

Theorem 5 (Safeness & minimality theorem). *Given a rewrite system h satisfying $\longrightarrow_{\mathsf{safecut}} \subseteq \longrightarrow_{\mathsf{h}} \subseteq \longrightarrow_{\mathsf{mincut}}$, suppose that we have:*

- *the strong simulation of $\longrightarrow_{\mathsf{mincut}} \setminus \longrightarrow_{\mathsf{h}}$ in a strongly normalising calculus, through a total relation $\mathcal{H}$*
- *the weak simulation of $\longrightarrow_{\mathsf{h}}$ through $\mathcal{H}$*
- *the strong normalisation of $\longrightarrow_{\mathsf{h}}$.*

Then $\longrightarrow_{\mathsf{cut}}$ is strongly normalising.

A naive attempt would be to take $\mathsf{h} = \mathsf{safecut}$, which terminates by Theorem 4(2). Unfortunately, this situation is too coarse, that is to say, the relation $\longrightarrow_{\mathsf{h}}$ is too small so that $\longrightarrow_{\mathsf{mincut}} \setminus \longrightarrow_{\mathsf{h}}$ is too big to be strongly simulated. Hence, in order to define h, we use the safeness criterion in a more subtle way, that is, we define $\mathsf{h} = \mathsf{safeB}, \mathsf{minx}$.

Among the conditions to apply Theorem 5, we first prove the third one, i.e. the strong normalisation of $\mathsf{safeB}, \mathsf{minx}$. For this we give a technical definition. The idea is to distinguish a class of terms with cut-constructors, reflecting the restrictions on permutations of two cuts in the rules (Perm_1) and (Perm_2).

[3] In both rules we could require $M \longrightarrow_{\mathsf{h}} N$ to be a root reduction so that M *is* the redex, but $\longrightarrow_{\mathsf{safeh}}$ and $\longrightarrow_{\mathsf{minh}}$ would be the same as they are.

Definition 2 (Application term). *A $\lambda G3$-term of the form $[M/x]N$ is called an* application term *if N is one of the forms: $[P/w]\langle \underline{x}M'/y\rangle N'$, $\langle \underline{x}M'/y\rangle N'$ and $[\langle \underline{x}M'/y\rangle N'/z]\langle \underline{z}M''/w\rangle N''$ where x occurs only once in N.*

Lemma 1. *If $[M/x]N$ is an application term and $N \longrightarrow_{cut} N'$, then $[M/x]N'$ is also an application term.*

Proof. It suffices to check each case. □

Next we briefly recall the lexicographic path ordering. For a more detailed description and proofs, the reader is referred to, e.g. [14,2].

Definition 3 (Lexicographic path ordering). *Let $\succ$ be a transitive and irreflexive ordering on the set of function symbols in a first-order signature, and let $s \equiv f(s_1, \ldots, s_m)$ and $t \equiv g(t_1, \ldots, t_n)$ be terms over the signature. Then $s >_{\mathsf{lpo}} t$, if one of the following holds:*

1. *$s_i \equiv t$ or $s_i >_{\mathsf{lpo}} t$ for some $i = 1, \ldots, m$,*
2. *$f \succ g$ and $s >_{\mathsf{lpo}} t_j$ for all $j = 1, \ldots, n$,*
3. *$f \equiv g$, $s >_{\mathsf{lpo}} t_j$ for all $j = 1, \ldots, n$, and $s_1 \equiv t_1, \ldots, s_{i-1} \equiv t_{i-1}, s_i >_{\mathsf{lpo}} t_i$ for some $i = 1, \ldots, m$.*

Theorem 6. *$>_{\mathsf{lpo}}$ is well-founded if and only if $\succ$ is well-founded.*

Now we encode λG3-terms into a first-order syntax given by the following ordered infinite signature:

$$\mathsf{sub}(\_,\_) \succ \mathsf{app}(\_,\_) \succ \mathsf{ii}(\_,\_) \succ \mathsf{i}(\_) \succ \mathsf{c}^M$$

where for every $M \in \mathsf{SN}^{\mathsf{cut}}$ there is a constant c^M. Those constants are all below $\mathsf{i}(\_)$, and the precedence between them is given by $\mathsf{c}^M \succ \mathsf{c}^N$ if $M \longrightarrow^+_{\mathsf{cut}} N$ or $M \sqsupset N$. Then the precedence relation is well-founded, and so $>_{\mathsf{lpo}}$ induced on the first-order terms is also well-founded. The encoding aforementioned is given in Figure 3.

$\overline{M}$	$:=$	c^M	if $M \in \mathsf{SN}^{\mathsf{cut}}$
otherwise			
$\overline{\lambda x.M}$	$:=$	$\mathsf{i}(\overline{M})$	
$\overline{\langle xM/y\rangle N}$	$:=$	$\mathsf{ii}(\overline{M}, \overline{N})$	
$\overline{[M/x]N}$	$:=$	$\mathsf{app}(\overline{M}, \overline{N})$	if $[M/x]N$ is an application term
$\overline{[M/x]N}$	$:=$	$\mathsf{sub}(\overline{M}, \overline{N})$	otherwise

Fig. 3. Encoding of λG3 into a first-order syntax

Lemma 2. *If $M \longrightarrow_{safeB,minx} M'$ then $\overline{M} >_{\mathsf{lpo}} \overline{M'}$. Hence, $\longrightarrow_{safeB,minx}$ is strongly normalising.*

Proof. By induction on the derivation of the reduction step. If $M \equiv [P/x]N$ is an application term and $N \longrightarrow_{\mathsf{safeB,minx}} N'$, then we use Lemma 1. (A detailed proof can be found in [18].) □

3.2 Simulation in λI

Now we have to find a strongly normalising calculus and a total relation $\mathcal{H}$ to strongly simulate $\longrightarrow_{\mathsf{mincut}} \setminus \longrightarrow_{\mathsf{h}}$ therein. Since a simple simulation in λ-calculus fails we use instead the λI-calculus of [19], based on earlier work by [6,24]. For instance, the technique works for proving PSN of the explicit substitution calculus $\lambda\mathsf{lxr}$ of [15]. We refer the reader to [27,34] for a survey on different techniques based on the λI-calculus to infer normalisation properties.

On the one hand, λI extends the syntax of λ-calculus with a "memory operator" so that, instead of being thrown away, a term U can be retained and carried along in a construct $[-, U]$. With this operator, those bodies of substitutions are encoded that would otherwise disappear, as explained above. On the other hand, λI restricts λ-abstractions to variables that have at least one free occurrence, so that β-reduction never erases its argument.

Definition 4 (Grammar of λI). *The set ΛI of terms of the λI-calculus of [19] is defined by the following grammar:*

$$T, U ::= x \mid \lambda x.T \mid T\ U \mid [T, U]$$

with the additional restriction that every abstraction $\lambda x.T$ satisfies $x \in \mathsf{FV}(T)$.

The following property is straightforward by induction on terms.

Lemma 3 (Stability under substitution [19]).
If $T, U \in \Lambda I$, then $\left\{{}^{U}\!/_{x}\right\}T \in \Lambda I$.

Definition 5 (Reduction system of λI). *The reduction rules are:*

$$\begin{array}{ll} (\beta) & (\lambda x.T)\ U \to \left\{{}^{U}\!/_{x}\right\}T \\ (\pi) & [T, U]\ T' \to [T\ T', U] \end{array}$$

The following remark is straightforward [19]:

Remark 2. If $T \longrightarrow_{\beta,\pi} T'$ then $\mathsf{FV}(T) = \mathsf{FV}(T')$ and $\left\{{}^{T}\!/_{x}\right\}U \longrightarrow^{+}_{\beta,\pi} \left\{{}^{T'}\!/_{x}\right\}U$ provided that $x \in \mathsf{FV}(U)$.

Performing a simulation in λI requires the encoding to be non-deterministic, i.e. we define a relation $\mathcal{H}$ between $\lambda\mathsf{G3}$ and λI, and the reason for this is that, since the reductions in λI are non-erasing reductions, we need to add this memory operator at random places in the encoding, using such a rule:

$$\frac{M\ \mathcal{H}\ T}{M\ \mathcal{H}\ [T, U]}\ U \in \Lambda I$$

The reduction relation of $\lambda\mathsf{G3}$ must then satisfy the hypotheses of Theorem 5. Namely, $\longrightarrow_{\mathsf{mincut}} \setminus \longrightarrow_{\mathsf{h}}$ should be strongly simulated by $\longrightarrow_{\beta,\pi}$ through $\mathcal{H}$, and $\mathsf{safeB}, \mathsf{minx}$ should be weakly simulated by $\longrightarrow_{\beta,\pi}$ through $\mathcal{H}$.

The relation $\mathcal{H}$ between $\lambda\mathsf{G3}$-terms and λI-terms is inductively defined in Figure 4.

$$\frac{}{x \mathrel{\mathcal{H}} x} \qquad \frac{M \mathrel{\mathcal{H}} T}{\lambda x.M \mathrel{\mathcal{H}} \lambda x.T}\, x \in \mathsf{FV}(T) \qquad \frac{M \mathrel{\mathcal{H}} U \quad N \mathrel{\mathcal{H}} T}{\langle xM/y\rangle N \mathrel{\mathcal{H}} \left\{{}^{x\ U}/_{y}\right\} T}\, y \in \mathsf{FV}(T)$$

$$\frac{M \mathrel{\mathcal{H}} T}{M \mathrel{\mathcal{H}} [T,U]}\, U \in \Lambda I \qquad \frac{M \mathrel{\mathcal{H}} U \quad N \mathrel{\mathcal{H}} T}{[M/x]N \mathrel{\mathcal{H}} \left\{{}^{U}/_{x}\right\} T}\, x \in \mathsf{FV}(T) \vee M \in \mathsf{SN}^{\mathsf{cut}}$$

Fig. 4. Relation between λG3 & λI

It satisfies the following properties:

Lemma 4. *If $M \mathrel{\mathcal{H}} T$, then*

1. $\mathsf{FV}(M) \subseteq \mathsf{FV}(T)$
2. $T \in \Lambda I$
3. $x \notin \mathsf{FV}(M)$ *and* $U \in \Lambda I$ *implies* $M \mathrel{\mathcal{H}} \left\{{}^{U}/_{x}\right\} T$
4. $\{{}^{y}/_{x}\} M \mathrel{\mathcal{H}} \{{}^{y}/_{x}\} T$.

Theorem 7 (Simulation in λI). *Suppose $M \mathrel{\mathcal{H}} T$.*

1. *If $M \longrightarrow_{\mathsf{minB}} N$ is unsafe then there exists U such that $N \mathrel{\mathcal{H}} U$ and $T \longrightarrow^{+}_{\beta\pi} U$.*
2. *If $M \longrightarrow_{\mathsf{minB}} N$ is safe then there exists U such that $N \mathrel{\mathcal{H}} U$ and $T \longrightarrow^{*}_{\beta\pi} U$.*
3. *If $M \longrightarrow_{\mathsf{minx}} N$ then there exists U such that $N \mathrel{\mathcal{H}} U$ and $T \longrightarrow^{*}_{\pi} U$.*

Proof. By induction on the derivation of the reduction step, by case analysis for root reduction. Indeed, for root-reduction, remember that we only simulate minimal reductions. Hence, when reducing a redex, all its subterms are in $\mathsf{SN}^{\mathsf{cut}}$, so the side-condition in the encoding of the cut-constructor is thus satisfied.

For the contextual closure, we have to ensure that, in the first of the above three points, the one reduction step that must take place is preserved through the inductive argument. This comes from the assumption that the reduction is unsafe, which ensures that, in the side-condition $x \in \mathsf{FV}(T) \vee M \in \mathsf{SN}^{\mathsf{cut}}$, it must be true that $x \in \mathsf{FV}(T)$.

A more detailed proof can be found in [18]. □

3.3 Concluding the Proof

Finally, we need the fact that every term M of λG3 that we wish to prove strongly normalising can be encoded into a strongly normalising term of λI, to start off the simulations. The following method works:

1. Encode the term M as a strongly normalising λ-term t, such that no subterm is lost, i.e. *not* using implicit substitutions.
2. Using a translation i from λ-calculus to λI, introduced in this section, prove that $\mathsf{i}(t)$ reduces to one of the non-deterministic encodings of M in λI, that is, that there is a term T such that $M \mathrel{\mathcal{H}} T$ and $\mathsf{i}(t) \longrightarrow^{*}_{\beta,\pi} T$.

The technique is summarised in Figure 5.

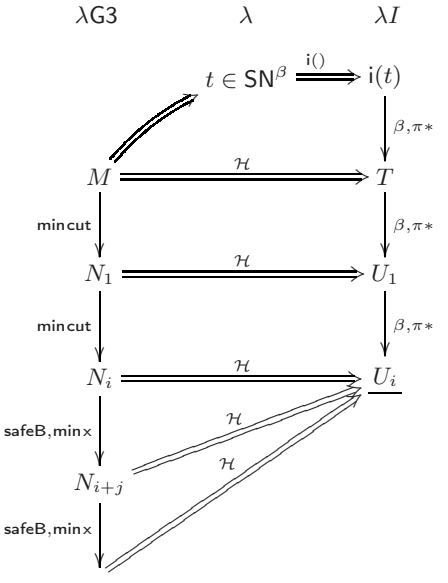

Fig. 5. The general technique to prove that $M \in \mathsf{SN}$

Definition 6 (Encoding of λG3 into λ-calculus). *We encode the λG3 into λ-calculus by slightly refining Gentzen's encoding as follows:*

$$
\begin{array}{lcll}
\mathcal{G}^{1\uparrow}(x) & := & x & \\
\mathcal{G}^{1\uparrow}(\lambda x.M) & := & \lambda x.\mathcal{G}^{1\uparrow}(M) & \\
\mathcal{G}^{1\uparrow}(\langle xM/y\rangle N) & := & \left\{{}^{x\ \mathcal{G}^{1\uparrow}(M)}\!/_{y}\right\}\mathcal{G}^{1\uparrow}(N) & \text{if } y \in \mathsf{FV}(N) \\
\mathcal{G}^{1\uparrow}(\langle xM/y\rangle N) & := & (\lambda y.\mathcal{G}^{1\uparrow}(N))\ (x\ \mathcal{G}^{1\uparrow}(M)) & \text{if } y \notin \mathsf{FV}(N) \\
\mathcal{G}^{1\uparrow}([M/x]N) & := & \left\{{}^{\mathcal{G}^{1\uparrow}(M)}\!/_{x}\right\}\mathcal{G}^{1\uparrow}(N) & \text{if } x \in \mathsf{FV}(N) \\
\mathcal{G}^{1\uparrow}([M/x]N) & := & (\lambda x.\mathcal{G}^{1\uparrow}(N))\ \mathcal{G}^{1\uparrow}(M) & \text{if } x \notin \mathsf{FV}(N)
\end{array}
$$

The reason why the above encoding is interesting for strong normalisation of some λG3-terms lies in two facts:

Lemma 5

1. *For the strong normalisation of typed terms:*
   If $\Gamma \vdash_{\lambda G3} M : A$ *then* $\Gamma \vdash_{\lambda} \mathcal{G}^{1\uparrow}(M) : A$
2. *For proving PSN:*
   $\mathcal{G}^{1\uparrow}(\mathcal{P}r(t)) = t$.

Proof. Straightforward inductions on M and t. □

Now we recall from [21,22] an encoding of λ-calculus into λI[4]:

Definition 7 (Encoding of λ-calculus into λI). *We encode the λ-calculus into λI as follows:*

$$\begin{array}{lcll} i(x) & := & x & \\ i(\lambda x.t) & := & \lambda x.i(t) & \text{if } x \in \mathsf{FV}(t) \\ i(\lambda x.t) & := & \lambda x.[i(t), x] & \text{if } x \notin \mathsf{FV}(t) \\ i(t\ u) & := & i(t)\ i(u) & \end{array}$$

The above encodings satisfy the following properties:

Lemma 6. *For any $\lambda G3$-term M, there is a λI-term T such that $M \mathcal{H} T$ and $i(\mathcal{G}^{1\uparrow}(M)) \longrightarrow^*_{\beta,\pi} T$.*

Proof. By induction on M. □

Theorem 8 ([21,22]). *For any λ-term t, if $t \in \mathsf{SN}^\beta$, then $i(t) \in \mathsf{SN}^{\beta,\pi}$.*

Hence we get

Corollary 1. *If $\mathcal{G}^{1\uparrow}(M) \in \mathsf{SN}^\beta$, then $M \in \mathsf{SN}^{cut}$.*

Proof. Suppose $\mathcal{G}^{1\uparrow}(M) \in \mathsf{SN}^\beta$. Then by Theorem 8, $\mathrm{i}(\mathcal{G}^{1\uparrow}(M)) \in \mathsf{SN}^{\beta,\pi}$, and so by Lemma 6, there is a λI-term T such that $M \mathcal{H} T$ and $T \in \mathsf{SN}^{\beta,\pi}$. Now apply Theorem 5 with Theorem 7 and Lemma 2. □

Finally this gives the two strong normalisation results:

Theorem 9 (Strong normalisation and PSN)
If $\Gamma \vdash_{\lambda G3} M : A$, or if $M = \mathcal{P}r(t)$ with $t \in \mathsf{SN}^\beta$, then $M \in \mathsf{SN}^{cut}$.

Proof. It suffices to combine Lemma 5 and Corollary 1. □

4 Related Work

In this section we discuss related work on strong normalisation of cut-elimination procedures. We focus on those cut-elimination procedures that have the ability to simulate β-reduction of the simply-typed λ-calculus.

Danos et al. [8,9] introduced strongly normalising cut-elimination procedures in sequent calculi for classical logic. The cut-elimination procedures include global proof transformations analogous to proof transformations in natural deduction. In rewrite systems for proof-terms, such cut-elimination procedures are implemented by reduction rules that use meta-operations like implicit

[4] Note that a similar encoding (without the case distinction for abstractions) can be found in [19]; unfortunately we have found it necessary to twist it to prove Theorem 8, which we have not found in the literature.

substitution in the λ-calculus. Urban [31] described a cut-elimination procedure for the classical sequent calculus in such a rewrite system. Many strong normalisation results of cut-elimination procedures with global proof transformations in the literature can be derived from Urban's result, in both classical and intuitionistic cases, including those for systems in the style of $\overline{\lambda}\mu\tilde{\mu}$ [7] (cf. e.g. [20]).

On the other hand, strong normalisation of cut-elimination procedures consisting of Gentzen-style local proof transformations requires us to use techniques from the field of calculi with explicit substitutions. Urban [31] proved strong normalisation of such a cut-elimination procedure using the technique of [5] and the strong normalisation result of his procedure with global proof transformations mentioned above. The cut-elimination procedure involves *labelled* cut, which are allowed to pass over usual cuts. In the present paper, cut-elimination uses only one kind of cut, and does not seem to be directly simulated by Urban's cut-elimination procedure. For example, the rule (Perm_1) corresponds to a permutation of labelled cuts, which is not included in Urban's reduction rules.

Another example of a cut-elimination procedure that consists of local proof transformations is the one by Dyckhoff and Urban [11] for Herbelin's sequent calculus [12]. Our method of proving strong normalisation works also for their system without using a simulation in λI. For the details, see [21,22].

Recently, Nakazawa [23] introduced a cut-elimination procedure for a standard intuitionistic sequent calculus, which is close to ours. The main difference between the two cut-elimination procedures is as follows. In his cut-elimination procedure, the redex $[\lambda z.P/x]\langle \underline{x}M/y\rangle N$ of the rule (B) is reduced to $[M/z][P/y]N$, while it is reduced to $[[M/z]P/y]N$ in our cut-elimination procedure. This difference corresponds to the order of applications of cuts in the resulting proofs. Strong normalisation of his cut-elimination procedure was proved by an inductive method as in [4], but it does not work for our rule (B) as explained in Section 6 of [23]. Another difference is that his cut-elimination procedure does not entirely follow Gentzen-style local steps; the cut-permutation rules of his cut-elimination procedure can be decomposed into two steps of ours (cf. Notes 3 and 4 of [23]).

5 Conclusion

We have proved strong normalisation of a cut-elimination procedure for a standard intuitionistic sequent calculus, by using the safeness and minimality technique and a simulation in λI, both of which are formalised in [21,22]. We have also established the PSN property of the type-free terms with respect to β-reduction through a Prawitz-style translation from the type-free λ-terms.

We consider our cut-elimination procedure for the intuitionistic sequent calculus as a canonical one, since it is strong normalising and confluent, consists of completely local steps (without an extra kind of cut), and can simulate β-reduction. For future work, it will be interesting to show strong normalisation of more liberal cut-elimination procedures such as the one in [16].

References

1. Arts, T., Giesl, J.: Termination of term rewriting using dependency pairs. Theoret. Comput. Sci. 236(1–2), 133–178 (2000)
2. Baader, F., Nipkow, T.: Term rewriting and all that. Cambridge University Press, Cambridge (1998)
3. Benaissa, Z., Briaud, D., Lescanne, P., Rouyer-Degli, J.: $\lambda\upsilon$, a calculus of explicit substitutions which preserves strong normalisation. J. Funct. Programming 6(5), 699–722 (1996)
4. Bloo, R.: Preservation of Termination for Explicit Substitution. PhD thesis, Technische Universiteit Eindhoven, IPA Dissertation Series 1997-05 (1997)
5. Bloo, R., Geuvers, H.: Explicit substitution: On the edge of strong normalization. Theoret. Comput. Sci. 211(1–2), 375–395 (1999)
6. Church, A.: The Calculi of Lambda Conversion. Princeton University Press, Princeton (1941)
7. Curien, P.-L., Herbelin, H.: The duality of computation. In: Proc. of the 5th ACM SIGPLAN Int. Conf. on Functional Programming (ICFP 2000), pp. 233–243. ACM Press, New York (2000)
8. Danos, V., Joinet, J.-B., Schellinx, H.: LKQ and LKT: Sequent calculi for second order logic based upon dual linear decompositions of classical implication. In: Girard, J.-Y., Lafont, Y., Regnier, L. (eds.) Proc. of the Work. on Advances in Linear Logic. London Math. Soc. Lecture Note Ser., vol. 222, pp. 211–224. Cambridge University Press, Cambridge (1995)
9. Danos, V., Joinet, J.-B., Schellinx, H.: A new deconstructive logic: Linear logic. J. of Symbolic Logic 62(3), 755–807 (1997)
10. Dyckhoff, R., Pinto, L.: Permutability of proofs in intuitionistic sequent calculi. Theoret. Comput. Sci. 212(1–2), 141–155 (1999)
11. Dyckhoff, R., Urban, C.: Strong normalization of Herbelin's explicit substitution calculus with substitution propagation. J. Logic Comput. 13(5), 689–706 (2003)
12. Herbelin, H.: A lambda-calculus structure isomorphic to Gentzen-style sequent calculus structure. In: Pacholski, L., Tiuryn, J. (eds.) CSL 1994. LNCS, vol. 933, pp. 61–75. Springer, Heidelberg (1995)
13. Howard, W.A.: The formulae-as-types notion of construction. In: Seldin, J.P., Hindley, J.R. (eds.) To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism, pp. 479–490. Academic Press, London (1980) (reprint of a manuscript written 1969)
14. Kamin, S., Lévy, J.-J.: Attempts for generalizing the recursive path orderings. Handwritten paper. University of Illinois (1980)
15. Kesner, D., Lengrand, S.: Resource operators for λ-calculus. Inform. and Comput. 205(4), 419–473 (2007)
16. Kikuchi, K.: On a local-step cut-elimination procedure for the intuitionistic sequent calculus. In: Hermann, M., Voronkov, A. (eds.) LPAR 2006. LNCS (LNAI), vol. 4246, pp. 120–134. Springer, Heidelberg (2006)
17. Kikuchi, K.: Confluence of cut-elimination procedures for the intuitionistic sequent calculus. In: Cooper, S.B., Löwe, B., Sorbi, A. (eds.) CiE 2007. LNCS, vol. 4497, pp. 398–407. Springer, Heidelberg (2007)
18. Kikuchi, K., Lengrand, S.: Strong normalisation of cut-elimination that simulates β-reduction - long version, `http://www.lix.polytechnique.fr/~lengrand/Work/`

19. Klop, J.-W.: Combinatory Reduction Systems, Mathematical Centre Tracts, PhD Thesis, vol. 127, CWI, Amsterdam (1980)
20. Lengrand, S.: Call-by-value, call-by-name, and strong normalization for the classical sequent calculus. In: Gramlich, B., Lucas, S. (eds.) Post-proc. of the 3rd Int. Work. on Reduction Strategies in Rewriting and Programming (WRS 2003). ENTCS, vol. 86, Elsevier, Amsterdam (2003)
21. Lengrand, S.: Induction principles as the foundation of the theory of normalisation: concepts and techniques. Technical report, Université Paris 7 (March 2005), http://hal.ccsd.cnrs.fr/ccsd-00004358
22. Lengrand, S.: Normalisation & Equivalence in Proof Theory & Type Theory. PhD thesis, Université Paris 7 & University of St. Andrews (2006)
23. Nakazawa, K.: An isomorphism between cut-elimination procedure and proof reduction. In: Della Rocca, S.R. (ed.) TLCA 2007. LNCS, vol. 4583, pp. 336–350. Springer, Heidelberg (2007)
24. Nederpelt, R.: Strong Normalization in a Typed Lambda Calculus with Lambda Structured Types. PhD thesis, Eindhoven University of Technology (1973)
25. Pottinger, G.: Normalization as a homomorphic image of cut-elimination. Ann. of Math. Logic 12, 323–357 (1977)
26. Santo, J.E.: Revisiting the correspondence between cut elimination and normalisation. In: Welzl, E., Montanari, U., Rolim, J.D.P. (eds.) ICALP 2000. LNCS, vol. 1853, pp. 600–611. Springer, Heidelberg (2000)
27. Sørensen, M.H.B.: Strong normalization from weak normalization in typed lambda-calculi. Inform. and Comput. 37, 35–71 (1997)
28. Sørensen, M.H.B., Urzyczyn, P.: Lectures on the Curry-Howard Isomorphism. Studies in Logic and the Foundations of Mathematics, vol. 149. Elsevier, Amsterdam (2006)
29. Sørensen, M.H.B., Urzyczyn, P.: Strong cut-elimination in sequent calculus using Klop's ι-translation and perpetual reduction (available from the authors) (submitted for publication, 2007)
30. Troelstra, A.S., Schwichtenberg, H.: Basic Proof Theory, 2nd edn. Cambridge Tracts in Theoret. Comput. Sci., vol. 43. Cambridge University Press, Cambridge (2000)
31. Urban, C.: Classical Logic and Computation. PhD thesis, University of Cambridge (2000)
32. Urban, C., Bierman, G.M.: Strong normalisation of cut-elimination in classical logic. In: Girard, J.-Y. (ed.) TLCA 1999. LNCS, vol. 1581, pp. 365–380. Springer, Heidelberg (1999)
33. van Raamsdonk, F., Severi, P., Sørensen, M.H.B., Xi, H.: Perpetual reductions in λ-calculus. Inform. and Comput. 149(2), 173–225 (1999)
34. Xi, H.: Weak and strong beta normalisations in typed lambda-calculi. In: de Groote, P., Hindley, J.R. (eds.) TLCA 1997. LNCS, vol. 1210, pp. 390–404. Springer, Heidelberg (1997)
35. Zucker, J.: The correspondence between cut-elimination and normalization. Ann. of Math. Logic 7, 1–156 (1974)