Strong Normalisation for a Gentzen-like Cut-Elimination Procedure

C. Urban*

Department of Pure Mathematics and Mathematical Statistics,
University of Cambridge.
cu200@dpmms.cam.ac.uk

Abstract. In this paper we introduce a cut-elimination procedure for classical logic, which is both strongly normalising and consisting of local proof transformations. Traditional cut-elimination procedures, including the one by Gentzen, are formulated so that they only rewrite neighbouring inference rules; that is they use local proof transformations. Unfortunately, such local proof transformation, if defined naïvely, break the strong normalisation property. Inspired by work of Bloo and Geuvers concerning the $\lambda\mathtt{x}$-calculus, we shall show that a simple trick allows us to preserve this property in our cut-elimination procedure. We shall establish this property using the recursive path ordering by Dershowitz.

Keywords. Cut-Elimination, Classical Logic, Explicit Substitution, Recursive Path Ordering.

1 Introduction

Gentzen showed in his seminal paper [6] that all cuts can be eliminated from sequent proofs in LK and LJ. He not only proved that cuts can be eliminated, but also gave a simple procedure for doing so. This procedure consists of proof transformations, or cut-reductions, that do not eliminate all cuts from a proof immediately, but rather replace every instance of a cut with simpler cuts, and by iteration one eventually ends up with a cut-free proof. We refer to a proof transformation as being local, or *Gentzen-like*, if it only rewrites neighbouring inference rules, possibly by duplicating a subderivation. Most of the traditional cut-elimination procedures, including Gentzen's original procedure, consist of such local proof transformations.

In [11] and [12] three criteria for a cut-elimination procedure were introduced:

1. the cut-elimination procedure should *not* restrict the collection of normal forms reachable from a given proof in such a way that "essential" normal forms are no longer reachable,
2. the cut-elimination procedure should be *strongly normalising*, i.e., all possible reduction strategies should terminate, and
3. the cut-elimination procedure should allow cuts to pass over other cuts.

* I should like to thank Roy Dyckhoff for many helpful discussions. I am currently funded with a research fellowship from Corpus Christi College, Cambridge.

Owing to space restrictions we cannot defend these criteria here and refer the reader to [11, 12] where it is explained why they play an important rôle in investigating the computational content of classical logic.

The main purpose of this paper is to present a cut-elimination procedure that satisfies the three criteria *and* that consists of only Gentzen-like cut-reductions. At the time of writing, we are not aware of any other cut-elimination procedure fulfilling both demands. The problem with Gentzen-like cut-reductions is that they, if defined naïvely, break the strong normalisation property, as illustrated in the following example given in [2, 5]. Consider the following LK-proof.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A\vee A\vdash A,A}\vee_L}{A\vee A\vdash A}\,\mathbf{Contr}_R \qquad \dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A,A\vdash A\wedge A}\wedge_R}{A\vdash A\wedge A}\,\mathbf{Contr}_L}{A\vee A\vdash A\wedge A}\,\mathbf{Cut} \tag{1}$$

Using Gentzen-like proof transformations there are two possibilities for permuting the cut upwards: it can either be permuted upwards in the left proof branch or in the right proof branch. In both cases a subproof has to be duplicated. Taking the former possibility, we obtain the proof

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A\vee A\vdash A,A}\vee_L \qquad \dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A,A\vdash A\wedge A}\wedge_R}{A\vdash A\wedge A}\,\mathrm{Contr}_L}{A\vee A\vdash A,A\wedge A}\,\mathrm{Cut} \qquad \dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A,A\vdash A\wedge A}\wedge_R}{A\vdash A\wedge A}\,\mathrm{Contr}_L}{A\vee A\vdash A\wedge A,A\wedge A}\,\mathrm{Cut}}{A\vee A\vdash A\wedge A}\,\mathrm{Contr}_R$$

where two copies of the right subproof are created. Now permute the upper cut to the right, which gives the following proof.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A\vee A\vdash A,A}\vee_L \qquad \dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A\vee A\vdash A,A}\vee_L \qquad \dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A,A\vdash A\wedge A}\wedge_R}{A\vee A,A\vdash A,A\wedge A}\,\mathrm{Cut}}{A\vee A,A\vee A\vdash A,A,A\wedge A}\,\mathrm{Contr}_L}{A\vee A\vdash A,A,A\wedge A}\,\mathbf{Contr}_R \qquad \dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A,A\vdash A\wedge A}\wedge_R}{A\vdash A\wedge A}\,\mathbf{Contr}_L}{A\vee A\vdash A\wedge A,A\wedge A}\,\mathbf{Cut}}{A\vee A\vdash A\wedge A}\,\mathrm{Contr}_R$$

This proof contains an instance of the reduction applied in the first step (compare the rule names in bold face). Even worse, it is bigger than the proof with which we started, and so in effect we can construct infinite reduction sequences.

Another problem with Gentzen-like cut-reductions arises from the third criterion. If one introduces the following reduction rule, which allows a cut (Suffix 2) to pass over another cut (Suffix 1)

$$\dfrac{\dfrac{\dfrac{}{\ldots\vdash\ldots}\quad\dfrac{}{\ldots\vdash\ldots}}{\ldots\vdash\ldots}\,\mathrm{Cut}_1 \qquad \dfrac{}{\ldots\vdash\ldots}}{\ldots\vdash\ldots}\,\mathrm{Cut}_2 \longrightarrow \dfrac{\dfrac{\dfrac{}{\ldots\vdash\ldots}\quad\dfrac{}{\ldots\vdash\ldots}}{\ldots\vdash\ldots}\,\mathrm{Cut}_2 \qquad \dfrac{}{\ldots\vdash\ldots}}{\ldots\vdash\ldots}\,\mathrm{Cut}_1 \tag{2}$$

then clearly one loses the strong normalisation property—the reduct is again an instance of this rule, and one can loop by constantly applying this reduction. Thus a common restriction is to not allow a cut to pass over another cut in any circumstances. Unfortunately, this has several serious drawbacks, as noted in [4, 7]; it limits, for example, in the intuitionistic case the correspondence between cut-elimination and beta-reduction. In particular, strong normalisation of beta-reduction cannot be inferred from strong normalisation of cut-elimination. Therefore we shall introduce cut-reductions that avoid the infinite reduction sequence illustrated in (1), but allow cuts to pass over other cuts without breaking the strong normalisation property.

Because of the conflicting demands of being very liberal (e.g. allowing cuts to pass over other cuts) and at the same time preserving the strong normalisation property, such a cut-elimination procedure seems difficult to obtain. So rather surprisingly we found that if one adds to the usual cut-rule

$$\frac{\Gamma_1 \vdash \Delta_1, C \quad C, \Gamma_2 \vdash \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\ \mathrm{Cut}$$

the following two, referred to as *labelled* cut-rules,

$$\frac{\Gamma_1 \vdash \Delta_1, C \quad C, \Gamma_2 \vdash \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\ \overleftarrow{\mathrm{Cut}} \qquad \frac{\Gamma_1 \vdash \Delta_1, C \quad C, \Gamma_2 \vdash \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\ \overrightarrow{\mathrm{Cut}}$$

then one can define a cut-elimination procedure that satisfies the three criteria *and* that only consists of Gentzen-like cut-reductions.

Reconsider the proof given in (1). There the infinite reduction sequence could be constructed by permuting cuts into alternating directions. Clearly, this reduction sequence can be avoided if commuting cuts have to be permuted into only one direction. (A cut is said to be a *logical* cut when both cut-formulae are introduced by axioms or logical inference rules; otherwise the cut is said to be a *commuting* cut.) Furthermore, instead of the cut-reduction shown in (2), we can introduce the following cut-reduction

$$\dfrac{\dfrac{\dots \vdash \dots \quad \dots \vdash \dots}{\dots \vdash \dots}\ \mathrm{Cut} \quad \dots \vdash \dots}{\dots \vdash \dots}\ \overleftarrow{\mathrm{Cut}} \longrightarrow \dfrac{\dfrac{\dots \vdash \dots \quad \dots \vdash \dots}{\dots \vdash \dots}\ \overleftarrow{\mathrm{Cut}} \quad \dots \vdash \dots}{\dots \vdash \dots}\ \mathrm{Cut}$$

which allows cuts to pass over other cuts, but which does not break the strong normalisation property—the reduction cannot be applied to the reduct.

Although the "trick" with the labelled cuts seems to be trivial, the corresponding strong normalisation proof is rather non-trivial (mainly because we allow cuts to pass over other cuts). To prove this property, we shall make use of a technique developed in [1]. This technique appeals to the recursive path ordering theorem by Dershowitz. Our proof is more difficult than the one given in [4], which also appeals to the recursive path ordering theorem, because we allow, as mentioned above, cuts to pass over other cuts. To be able to present our proof in a manageable form, we found it extremely useful to annotate sequent proofs with terms. In consequence, our contexts are sets of (label,formula) pairs, as in type theory, and *not* multisets, as in LK or LJ.

The paper is organised as follows. In Section 2 our sequent calculus and the corresponding term annotations will be given. To save space, we shall restrict our attention in this paper to the $\wedge$-fragment of classical logic, but it should be emphasised that the strong normalisation result for cut-elimination may be obtained for all connectives by a simple adaptation of the proof we shall give. The cut-elimination procedure will be defined in Section 3. A comparison with the $\lambda\mathtt{x}$-calculus will be given in Section 4. In Section 5 we shall describe the proof of strong normalisation, and conclude and give suggestions for further work in Section 6.

2 Sequent Calculus, Terms and Typing Judgements

In this section we shall introduce our sequent calculus for classical logic. As mentioned earlier, we shall restrict our attention to the $\wedge$-fragment. Thus the formulae are given by the grammar

$$B ::= A \mid B \wedge B$$

in which A ranges over propositional symbols.

Our sequents contain two contexts—an *antecedent* and a *succedent*—both of which are sets of (label,formula) pairs. As we shall see, the use of sets allows us to define the sequent calculus so that the structural rules, i.e., weakening and contraction, are completely implicit in the form of the logical inference rules. Since there are two sorts of contexts, it will be convenient to separate the labels into *names* and *co-names*; in what follows a, b, c, ... will stand for co-names and similarly ..., x, y, z for names. Thus, antecedents are built up by (name,formula) pairs and succedents by (co-name,formula) pairs. We shall employ some shorthand notation for contexts: rather than writing, for example, $\{(x, B), (y, C), (z, D)\}$, we shall simply write $x : B, y : C, z : D$.

Whereas in LK the sequents consists of an antecedent and succedent only, in our sequent calculus the sequents have another component: a *term*. Terms encode the structure of sequent proofs and thus allow us to define a complete cut-elimination procedure as a term rewriting system. The set of raw terms, $\mathcal{R}_\wedge$, is defined by the grammar

$$\begin{array}{rll}
M, N ::= & \mathsf{Ax}(x, a) & \text{Axiom} \\
\mid & \mathsf{And}_R(\langle a{:}B\rangle M, \langle b{:}C\rangle N, c) & \text{And-R} \\
\mid & \mathsf{And}^i_L((x{:}B)M, y) & \text{And-L}_i \qquad (i = 1, 2) \\
\mid & \mathsf{Cut}(\langle a{:}B\rangle M, (x{:}B)N) & \text{Cut} \\
\mid & \mathsf{C\underleftarrow{ut}}(\langle a{:}B\rangle M, (x{:}B)N) & \text{Cut with label `}\leftarrow\text{'} \\
\mid & \mathsf{C\underrightarrow{ut}}(\langle a{:}B\rangle M, (x{:}B)N) & \text{Cut with label `}\rightarrow\text{'}
\end{array}$$

where x, y are taken from a set of names and a, b, c from a set of co-names; B and C are types (formulae). In a term we use round brackets to signify that a name becomes bound and angle brackets that a co-name becomes bound. In what follows we shall often omit the types on the bindings for brevity, regard terms as equal up to alpha-conversions and adopt a Barendregt-style convention

for the names and co-names. These conventions are standard in term rewriting. Notice however that names and co-names are not the same notions as a variable in the lambda-calculus: whilst a term can be substituted for a variable, a name or a co-name can only be "renamed". Rewriting a name x to y in a term M is written as $M[x \mapsto y]$, and similarly rewriting a co-name a to b is written as $M[a \mapsto b]$. The routine formalisation of these rewriting operations is omitted. In our proof it will be useful to have the following notions: a term is said to be *labelled* provided its top-most term constructor is either $\mathsf{C}\overleftarrow{\mathsf{ut}}$ or $\mathsf{C}\overrightarrow{\mathsf{ut}}$; otherwise the term is said to be *unlabelled*; a term is said to be *completely unlabelled* provided all subterms are unlabelled. Other useful notions are as follows.

- A term, M, *introduces* the name z or co-name c iff M is of the form

  for z: $\mathsf{Ax}(z, c)$, $\mathsf{And}^i_L((x)S, z)$ for c: $\mathsf{Ax}(z, c)$, $\mathsf{And}_R(\langle a\rangle S, \langle b\rangle T, c)$

- A term, M, *freshly introduces* a name iff M introduces this name, but none of its proper subterms. In other words, the name must not be free in a proper subterm, just in the top-most term constructor. Similarly for co-names.

We can now formally introduce sequents, or *typing judgements*. They are of the form $\Gamma \triangleright M \triangleright \Delta$ with Γ being an antecedent, M a term and Δ a succedent. Henceforth we shall be interested in only *well-typed* terms; this means those for which there are two contexts, Γ and Δ, such that $\Gamma \triangleright M \triangleright \Delta$ holds given the inference rules in Figure 1. We shall write $\mathcal{T}_\wedge$ for the set of well-typed terms.

Whilst the structural rules are *implicit* in our sequent calculus, i.e., the calculus has fewer inference rules, there are a number of subtleties concerning contexts. First, we assume the convention that a context is ill-formed, if it contains more than one occurrence of a name or co-name. For example the antecedent $x{:}B, x{:}C$ is not allowed. Hereafter, this will be referred to as the context convention, and it will be assumed that all inference rules respect this convention.

Second, we have the following conventions for the commas in Figure 1: a comma in a conclusion stands for set union and a comma in a premise stands for *disjoint* set union. Consider for example the $\wedge_{L_i}$-rule. This rule introduces the (name,formula) pair $y : B_1 \wedge B_2$ in the conclusion, and consequently, y is a free name in $\mathsf{And}^i_L((x)M, y)$. However, y can already be free in the subterm M, in which case $y{:}B_1 \wedge B_2$ belongs to Γ. We refer to this as an *implicit contraction*. Hence the antecedent of the conclusion of $\wedge_{L_i}$ is of the form $y{:}B_1 \wedge B_2 \oplus \Gamma$ where $\oplus$ denotes set union. Clearly, if the term $\mathsf{And}^i_L((x)M, y)$ freshly introduces y, then this antecedent is of the form $y{:}B_1 \wedge B_2 \otimes \Gamma$ where $\otimes$ denotes *disjoint* set union. Note that $x{:}B_i$ cannot be part of the conclusion: x becomes bound in the term. Thus the antecedent of the premise must be of the form $x{:}B_i \otimes \Gamma$.

There is one point worth mentioning in the cut-rules, because they are the only inference rules in our sequent calculus that do not share the contexts, but require that two contexts are joined on each side of the conclusion. Thus we take the cut-rule labelled with '$\leftarrow$', for example, to be of the form

$$\frac{\Gamma_1 \triangleright M \triangleright \Delta_1 \otimes a{:}B \qquad x{:}B \otimes \Gamma_2 \triangleright N \triangleright \Delta_2}{\Gamma_1 \oplus \Gamma_2 \triangleright \overleftarrow{\mathsf{Cut}}(\langle a\rangle M, (x)N) \triangleright \Delta_1 \oplus \Delta_2}\ \overleftarrow{\mathrm{Cut}}\,.$$

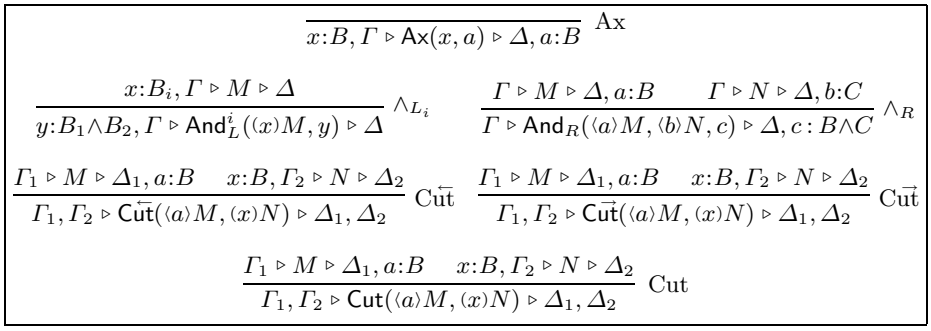

Fig. 1. Term assignment for sequent proofs in the $\wedge$-fragment of classical logic.

In effect, this rule is only applicable, if it does not break the context convention, which can always be achieved by renaming some labels appropriately. Notice that we do not require that cut-rules have to be "fully" multiplicative: the Γ_i's (respectively the Δ_j's) can share some formulae.

3 Cut-Reductions

We are now ready to define our Gentzen-like cut-elimination procedure. For this we shall introduce four sorts of cut-reduction, each of which is assumed to be closed under context formation. This is a standard convention in term rewriting. The first sort of cut-reduction, written $\xrightarrow{l}$, deals with logical cuts.

Logical Reductions:

1. $\mathsf{Cut}(\langle b\rangle \mathsf{And}_R(\langle a_1\rangle M_1, \langle a_2\rangle M_2, b), (y)\mathsf{And}^i_L((x)N, y)) \xrightarrow{l} \mathsf{Cut}(\langle a_i\rangle M_i, (x)N)$
   if $\mathsf{And}_R(\langle a_1\rangle M_1, \langle a_2\rangle M_2, b)$ and $\mathsf{And}^i_L((x)N, y)$ freshly introduce b and y
2. $\mathsf{Cut}(\langle a\rangle M, (x)\mathsf{Ax}(x, b)) \xrightarrow{l} M[a \mapsto b]$
   if M freshly introduces a
3. $\mathsf{Cut}(\langle a\rangle \mathsf{Ax}(y, a), (x)M) \xrightarrow{l} M[x \mapsto y]$
   if M freshly introduces x

As can be seen, these cut-reductions are restricted so that they are applicable only if the immediate subterms of the cuts freshly introduce the names and co-names corresponding to the cut-formulae. Without this restriction bound names or bound co-names might become free during cut-elimination, as demonstrated in [11, 12]. Note that in Reduction 2 (resp. 3) it is permitted that b (resp. y) is free in M.

The next sort of cut-reduction applies to commuting cuts, that means to those where at least one immediate subterm of the cut does not freshly introduce the name or co-name of the cut-formula.

Commuting Reductions:

5. $\mathsf{Cut}(\langle a\rangle M, (x)N) \xrightarrow{c} \overleftarrow{\mathsf{Cut}}(\langle a\rangle M, (x)N)$
   if M does not freshly introduce a and is unlabelled, *or*
6. $\mathsf{Cut}(\langle a\rangle M, (x)N) \xrightarrow{c} \overrightarrow{\mathsf{Cut}}(\langle a\rangle M, (x)N)$
   if N does not freshly introduce x and is unlabelled.

A point to note is that Reductions 5 and 6 may be applicable at the same time. Take for example the term $\mathsf{Cut}(\langle a\rangle \mathsf{Ax}(x,b), (y)\mathsf{Ax}(z,c))$, which can reduce to either $\overleftarrow{\mathsf{Cut}}(\langle a\rangle \mathsf{Ax}(x,b), (y)\mathsf{Ax}(z,c))$ or $\overrightarrow{\mathsf{Cut}}(\langle a\rangle \mathsf{Ax}(x,b), (y)\mathsf{Ax}(z,c))$—the choice to which term it reduces is not specified. Therefore, our cut-elimination procedure is non-deterministic.

Once a cut is "labelled" by Reduction 5 or 6, then cut-reductions written as $\xrightarrow{x}$ apply (see Figure 2). Each of them pushes labelled cuts inside the subterms until they reach a place where the cut-formula is introduced. However care needs to be taken when applying an $\xrightarrow{x}$-reduction to ensure that no name or co-name clash occurs. This can always be achieved by appropriate alpha-conversions, and we shall assume that these conversions are done implicitly.

It is worthwhile to comment on the reductions $\xrightarrow{c}$ and $\xrightarrow{x}$. We required in Reduction 5 (similarly in 6) that the term M is unlabelled, i.e., the top-most term constructor is not $\overleftarrow{\mathsf{Cut}}$ or $\overrightarrow{\mathsf{Cut}}$. This restriction is to avoid certain reduction sequences. Suppose M and N are cut-free, and assume the term $\mathsf{Cut}(\langle a\rangle M, (x)N)$ is a logical cut. Furthermore assume c is not free in this term. Then consider the reduction sequence

$$\begin{array}{rl} \overleftarrow{\mathsf{Cut}}(\langle c\rangle \mathsf{Cut}(\langle a\rangle M, (x)N), (y)P) \xrightarrow{x} & \mathsf{Cut}(\langle a\rangle \overleftarrow{\mathsf{Cut}}(\langle c\rangle M, (y)P), (x)\overleftarrow{\mathsf{Cut}}(\langle c\rangle N, (y)P)) \\ \xrightarrow{c} & \overleftarrow{\mathsf{Cut}}(\langle a\rangle \overleftarrow{\mathsf{Cut}}(\langle c\rangle M, (y)P), (x)\overleftarrow{\mathsf{Cut}}(\langle c\rangle N, (y)P)) \\ \xrightarrow{x}{}^{+} & \overleftarrow{\mathsf{Cut}}(\langle a\rangle M, (x)N) \end{array}$$

where the logical cut has become labelled ($\xrightarrow{c}$-reduction), because another cut passed over it (first $\xrightarrow{x}$-reduction). While this reduction is harmless with respect to strong normalisation (this cut becomes a logical cut again), it causes the strong normalisation proof to be much harder. To save space, we thus exclude reduction sequences in which a logical cut becomes labelled, and the side-conditions in Reduction 5 and 6 are doing just that.

Another point worth mentioning is that the first and second rule in Figure 2 (similarly the fourth and fifth) can be replaced with the reduction

$$\overleftarrow{\mathsf{Cut}}(\langle c\rangle \mathsf{Ax}(x,c), (y)P) \longrightarrow P[y \mapsto x] \tag{3}$$

which is equally effective, in that all cut-rules are eliminable from a proof. However, this reduction has subtle defect, as explained in [11, 12]. Consider a term N in which x is not free and a term P in which b is not free. We would expect that from $\overrightarrow{\mathsf{Cut}}(\langle a\rangle N, (x)\overleftarrow{\mathsf{Cut}}(\langle b\rangle M, (y)P))$ and $\overleftarrow{\mathsf{Cut}}(\langle b\rangle \overrightarrow{\mathsf{Cut}}(\langle a\rangle N, (x)M), (y)P)$ the same collection of normal forms can be reached (the order of "independent" labelled cuts should not matter). Unfortunately, using the rule in (3) this does *not* hold. Therefore we have formulated the $\xrightarrow{x}$-reductions so that the the order of labelled cuts—as long as they are "independent"—is irrelevant with respect to

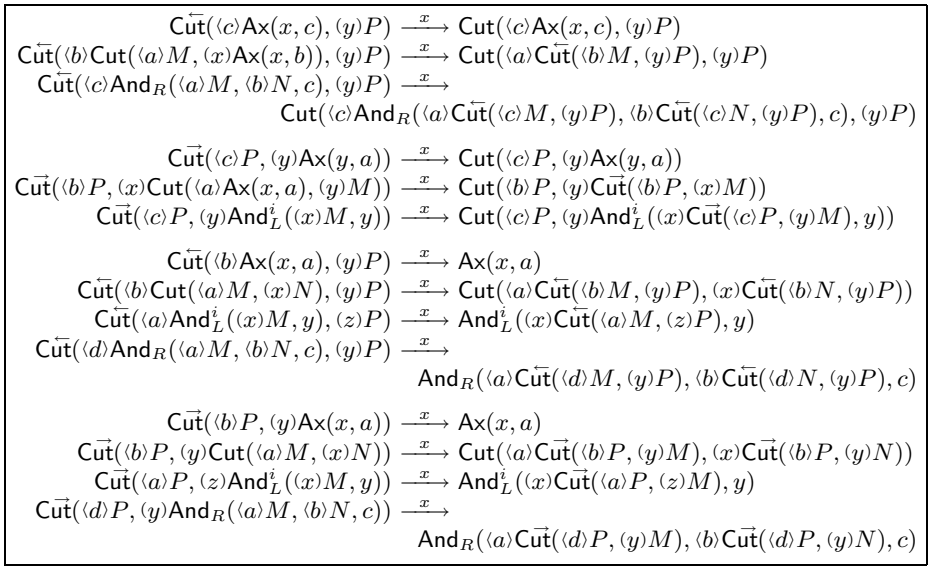

Fig. 2. Cut-reductions for labelled cuts.

which normal forms are reachable. This is an important property for analysing the computational content of classical proofs [8].

The last sort of cut-reduction, named *garbage reduction*, deals with labelled cuts whose name or co-name of the cut-formula is not free in the corresponding subterm. In LK this corresponds to a cut on a weakened formula.

Garbage Reductions:

7. $\mathsf{C\underline{u}t}(\langle a\rangle M,(x)N) \xrightarrow{gc} M$ if a is not a free co-name in M
8. $\mathsf{C\underline{u}t}(\langle a\rangle M,(x)N) \xrightarrow{gc} N$ if x is not a free name in N

We are now ready to define our Gentzen-like cut-elimination procedure. Since we annotated terms to our sequent proofs, we can define it as a term rewriting system.

Definition 1 (Gentzen-like Cut-Elimination Procedure). The Gentzen-like cut-elimination procedure is the term rewriting system $(\mathcal{T}_\wedge, \xrightarrow{loc})$ where:

- $\mathcal{T}_\wedge$ is the set of terms well-typed by the rules shown in Figure 1, and
- $\xrightarrow{loc}$ consists of the reduction rules for logical, commuting and labelled cuts as well as the garbage reductions; that is

$$\xrightarrow{loc} \stackrel{\text{def}}{=} \xrightarrow{l} \cup \xrightarrow{c} \cup \xrightarrow{x} \cup \xrightarrow{gc} .$$

Notice that by assumption all reductions are closed under context formation. The completeness of $\xrightarrow{loc}$ is simply the fact that every term beginning with a cut matches at least one left-hand side of the reduction rules. So each irreducible term is cut-free. We shall however omit a proof of this fact. The theorem for

which we are going to give a proof for is as follows, but we delay the proof until Section 5.

Theorem 1. For all terms in $\mathcal{T}_\wedge$ the reduction $\xrightarrow{loc}$ is strongly normalising.

As said earlier, this theorem can be generalised to include all connectives, and our proof can be easily adapted to the more general case.

4 Comparison with Explicit Substitution Calculi

There is a close correspondence between our cut-elimination procedure and explicit substitution calculi, as we shall illustrate in this section.

Explicit substitution calculi have been developed to internalise the substitution operation—a meta-level operation on lambda-terms—arising from beta-reductions. For example in $\lambda\mathtt{x}$ [10], the beta-reduction $(\lambda x.M)N \xrightarrow{\beta} M[x := N]$ is replaced by the reduction $(\lambda x.M)N \xrightarrow{b} M\langle x := N\rangle$ where the reduct contains a new syntactic constructor. The following reduction rules apply to this constructor.

$$\begin{aligned} y\langle x := P\rangle &\xrightarrow{x} P \ \text{ if } x \equiv y \ \text{ otherwise } \ y \\ (\lambda y.M)\langle x := P\rangle &\xrightarrow{x} \lambda y.\, M\langle x := P\rangle \\ (MN)\langle x := P\rangle &\xrightarrow{x} M\langle x := P\rangle\, N\langle x := P\rangle \end{aligned}$$

Similarly, our labelled cuts internalise a proof substitution introduced in [11, 12]. This substitution operation is written as $M\{a := (x)N\}$ and $N\{x := \langle a\rangle M\}$ where M and N belong to $\mathcal{TU}_\wedge$ that is defined as the set of terms well-typed by the typing rules given in Figure 1 excluding the rules for labelled cuts. Thus $\mathcal{TU}_\wedge$ consists of well-typed but completely unlabelled terms, and clearly, we have $\mathcal{TU}_\wedge \subset \mathcal{T}_\wedge$. In terms of the reductions given above the proof substitution can be defined as the juxtaposition of a $\xrightarrow{c}$-reduction and a series of $\xrightarrow{x}$-reductions, which need to be applied until no further $\xrightarrow{x}$-reduction is applicable (later we shall refer to such a term as x-normal form). Here we omit an inductive definition of the proof substitution, which can be found in [11, 12]. Using this proof substitution we can reformulate the reduction for commuting cuts as follows.

5'. $\mathsf{Cut}(\langle a\rangle M, (x)N) \xrightarrow{c'} M\{a := (x)N\}$ if M does not freshly introduce a, *or*

6'. $\mathsf{Cut}(\langle a\rangle M, (x)N) \xrightarrow{c'} N\{x := \langle a\rangle M\}$ if N does not freshly introduce x.

This leads to the following cut-elimination procedure, which satisfies the three criteria given in the introduction, but which is *not* Gentzen-like (the proof substitution is a "global" operation).

Definition 2 (Global Cut-Elimination Procedure). The cut-elimination procedure $(\mathcal{TU}_\wedge, \xrightarrow{gbl})$ is the term rewriting system where:

- $\mathcal{TU}_\wedge$ is the set of well-typed but completely unlabelled terms, and
- $\xrightarrow{gbl}$ consists of the reduction rules for logical and commuting cuts; that is

$$\xrightarrow{gbl} \stackrel{\text{def}}{=} \xrightarrow{l} \cup \xrightarrow{c'} .$$

A proof of strong normalisation for $(\mathcal{TU}_\wedge, \xrightarrow{gbl})$ is given in [11, 12]. There is no known technique that would give a strong normalisation result for $(\mathcal{T}_\wedge, \xrightarrow{loc})$ via a simple translation from $\mathcal{T}_\wedge$ to $\mathcal{TU}_\wedge$. This is similar to the situation with the lambda-calculus and $\lambda\mathtt{x}$: strong normalisation for the explicit substitution calculus does not follow directly from strong normalisation of the lambda-calculus. Indeed as shown in [9] explicit substitution calculi, if defined naïvely, may break the strong normalisation property. So the proof we shall present next is rather involved.

5 Proof of Strong Normalisation

In this section we shall give a proof for Theorem 1. In this proof we shall make use of the recursive path ordering by Dershowitz [3].

Definition 3 (Recursive Path Ordering). Let $s \equiv f(s_1, \ldots, s_m)$ and $t \equiv g(t_1, \ldots, t_n)$ be terms, then $s >^{rpo} t$ iff

	(i)	$s_i \geq^{rpo} t$ for some $i = 1, \ldots, m$	(subterm)
or	(ii)	$f \gg g$ and $s >^{rpo} t_j$ for all $j = 1, \ldots, n$	(decreasing heads)
or	(iii)	$f = g$ and $\{\!\{s_1, \ldots, s_m\}\!\} >^{rpo}_{mult} \{\!\{t_1, \ldots, t_n\}\!\}$	(equal heads)

where $\gg$ is a precedence defined over term constructors, $>^{rpo}_{mult}$ is the extension of $>^{rpo}$ to finite multisets and $\{\!\{\ldots\}\!\}$ stands for a multiset of terms; $\geq^{rpo}$ means $>^{rpo}$ or equivalent up to permutation of subterms.

The recursive path ordering theorem says that $>^{rpo}$ is well-founded iff the precedence of the term constructors, $\gg$, is well-founded. Unfortunately, two problems preclude a direct application of this theorem.

- First, the theorem requires a well-founded precedence for our term constructors. However, our reduction rules include the two reductions (written schematically)

$$\begin{aligned} \mathsf{Cut}(\_,\_) &\xrightarrow{c} \overleftarrow{\mathsf{Cut}}(\_,\_) \\ \overleftarrow{\mathsf{Cut}}(\mathsf{Cut}(\_,\_),\_) &\xrightarrow{x} \mathsf{Cut}(\overleftarrow{\mathsf{Cut}}(\_,\_), \overleftarrow{\mathsf{Cut}}(\_,\_)) \end{aligned}$$

  and consequently we have a cycle between Cut and $\overleftarrow{\mathsf{Cut}}$. In [1] a clever solution for an analogous problem in $\lambda\mathtt{x}$ was presented. We shall adapt this solution for our rewrite system. The essence of this solution is that we take into account (in a non-trivial way) that $(\mathcal{TU}_\wedge, \xrightarrow{gbl})$ is strongly normalising.
- The second problem arises from the fact that the recursive path ordering theorem applies only to first-order rewrite systems, i.e., no binding operations are allowed. In our term calculus however we have two binding operations: one for names and one for co-names. We solve this problem by introducing another term calculus, denoted by $\mathcal{H}$, for which we can apply this theorem, and then prove strong normalisation for $(\mathcal{T}_\wedge, \xrightarrow{loc})$ by translation.

The first important fact in our proof is that $\xrightarrow{x}$ is confluent, in contrast to $\xrightarrow{loc}$, which is clearly not.

Lemma 1. The reduction $\xrightarrow{x}$ is strongly normalising and confluent.

Proof. We can show the first part of the lemma by a simple calculation using the measure, $[\_]$, that is 1 for axioms and that is the sum of the measures of the subterms increased by 1 for $\mathsf{And}_R(\_,\_)$; similarly for $\mathsf{And}^i_L(\_)$ and $\mathsf{Cut}(\_,\_)$. For the labelled cuts we have:

$$[\vec{\mathsf{Cut}}(\langle a\rangle M, (x)N)] \stackrel{\text{def}}{=} ([M]+1)*(4[N]+1)$$
$$[\overleftarrow{\mathsf{Cut}}(\langle a\rangle M, (x)N)] \stackrel{\text{def}}{=} (4[M]+1)*([N]+1)$$

This gives $[M] > [N]$ whenever $M \xrightarrow{x} N$. Confluence of $\xrightarrow{x}$ follows from local confluence, which can be easily established, and strong normalisation. □

As a result, we can define the unique x-normal form of a term belonging to $\mathcal{T}_\wedge$.

Definition 4. The unique x-normal form of a term $M \in \mathcal{T}_\wedge$ is denoted by $|M|_x$.

By a careful case analysis we can show that for all $M \in \mathcal{T}_\wedge$ the x-normal form $|M|_x$ is an element $\mathcal{TU}_\wedge$, i.e., is well-typed and completely unlabelled. The details are omitted.

Next we shall prove that $\xrightarrow{x}$ correctly simulates the proof substitution operation of $\xrightarrow{gbl}$.

Lemma 2. For all $M, N \in \mathcal{T}_\wedge$ we have

(i) $|\overleftarrow{\mathsf{Cut}}(\langle a\rangle M, (y)N)|_x \equiv |M|_x\{a := (y)\,|N|_x\}$
(ii) $|\vec{\mathsf{Cut}}(\langle a\rangle N, (y)M)|_x \equiv |M|_x\{y := \langle a\rangle\,|N|_x\}$

Proof. We can show the lemma by induction on M in case M is completely unlabelled. We can then prove the lemma for all terms by a simple calculation, as illustrated next for (i): by uniqueness of the x-normal form we have that $|\overleftarrow{\mathsf{Cut}}(\langle a\rangle M, (y)N)|_x \equiv |\overleftarrow{\mathsf{Cut}}(\langle a\rangle\,|M|_x, (y)\,|N|_x)|_x$ and, because $|M|_x$ is completely unlabelled, this is $|M|_x\{a := (y)\,|N|_x\}$. □

Now we are in a position to show another important fact in our proof, namely that the $\xrightarrow{loc}$-reductions project onto $\xrightarrow{gbl}$-reductions.

Lemma 3. For all terms $M, N \in \mathcal{T}_\wedge$ if $M \xrightarrow{loc} N$ then $|M|_x \xrightarrow{gbl} |N|_x$.

Proof. By induction on the definition of $\xrightarrow{loc}$. □

As mentioned earlier, this lemma is not strong enough to prove strong normalisation of $\xrightarrow{loc}$. To prove this property we shall use a translation that maps every $\xrightarrow{loc}$-reduction onto a pair of terms belonging to the set $\mathcal{H}$, defined as follows.

Definition 5. Let $\mathcal{H}$ be the set of all terms generated by the grammar

$$M, N ::= \star \mid M \cdot_n N \mid M\langle N\rangle_n \mid \langle M\rangle_n N \mid (\!|M, N|\!) \mid (\!|M|\!)$$

where n is a natural number. The well-founded precedence $\gg$ is given by

$$\_ \cdot_{n+1} \_ \;\gg\; \langle\_\rangle_n\_\,,\ \_\langle\_\rangle_n \;\gg\; \_ \cdot_n \_ \;\gg\; \star,\ (\!|\_|\!),\ (\!|\_,\_|\!)\,.$$

To define the translation we shall use, as it turns out later, an alternative definition of the set $\mathcal{T}_\wedge$. This alternative definition is required in order to strengthen an induction hypothesis.

Definition 6. The set of *bounded* terms, $\mathcal{B}_\wedge$, consists of well-typed terms M wherby for every subterm N of the M the corresponding x-normal form, $|N|_x$, must be strongly normalising with respect to $\xrightarrow{gbl}$.

Clearly, we now have to show the fact that

Lemma 4. The set of bounded terms is closed under $\xrightarrow{loc}$-reductions.

Proof. By induction on the definition of $\xrightarrow{loc}$ using Lemma 3.

Next we define the translation from bounded terms to terms of $\mathcal{H}$.

Definition 7. The translation $\_ : \mathcal{B} \to \mathcal{H}$ is inductively defined by the clauses

$$\underline{\mathsf{Ax}(x,a)} \stackrel{\text{def}}{=} \star \qquad \underline{\mathsf{And}_R(\langle a\rangle S, \langle b\rangle T, c)} \stackrel{\text{def}}{=} (\!|\underline{S}, \underline{T}|\!) \qquad \underline{\mathsf{And}^i_L(\langle x\rangle S, y)} \stackrel{\text{def}}{=} (\!|\underline{S}|\!)$$

$$\begin{array}{lll}
\underline{\mathsf{Cut}(\langle a\rangle S, \langle x\rangle T)} \stackrel{\text{def}}{=} \underline{S} \cdot_l \underline{T} & l \stackrel{\text{def}}{=} & \textsc{maxred}_{gbl}(|\mathsf{Cut}(\langle a\rangle S, \langle x\rangle T)|_x) \\
\underline{\overleftarrow{\mathsf{Cut}}(\langle a\rangle S, \langle x\rangle T)} \stackrel{\text{def}}{=} \underline{S}\langle\underline{T}\rangle_m & m \stackrel{\text{def}}{=} & \textsc{maxred}_{gbl}(|\overleftarrow{\mathsf{Cut}}(\langle a\rangle S, \langle x\rangle T)|_x) \\
\underline{\overrightarrow{\mathsf{Cut}}(\langle a\rangle S, \langle x\rangle T)} \stackrel{\text{def}}{=} \langle\underline{S}\rangle_n \underline{T} & n \stackrel{\text{def}}{=} & \textsc{maxred}_{gbl}(|\overrightarrow{\mathsf{Cut}}(\langle a\rangle S, \langle x\rangle T)|_x)
\end{array}$$

where $\textsc{maxred}_{gbl}(|M|_x)$ denotes the number of steps of the longest $\xrightarrow{gbl}$-reduction sequence starting from the x-normal form of M. Clearly, this translation is well-defined since it is restricted to bounded terms. The next lemma will be applied when we need to compare labels of terms in $\mathcal{H}$.

Lemma 5. For all terms $M, N \in \mathcal{B}$ we have

(i) $\textsc{maxred}_{gbl}(|M|_x) \geq \textsc{maxred}_{gbl}(|N|_x)$, provided $M \xrightarrow{loc} N$.
(ii) $\textsc{maxred}_{gbl}(|M|_x) \geq \textsc{maxred}_{gbl}(|N|_x)$, provided N is an immediate subterm of M and M is unlabelled.
(iii) $\textsc{maxred}_{gbl}(|M|_x) > \textsc{maxred}_{gbl}(|N|_x)$, provided $M \xrightarrow{l} N$ or $M \xrightarrow{c} N$ on the outermost level.

Proof. (i) follows from Lemma 3; for (ii) note that all reductions which $|N|_x$ can perform can be performed by $|M|_x$; (iii) is by a simple calculation and the fact that the side conditions put on $\xrightarrow{c}$ ensures that $|M|_x \xrightarrow{gbl} |N|_x$. □

We shall now prove the (main) lemma, which relates a $\xrightarrow{loc}$-reduction to a pair of terms belonging to $\mathcal{H}$ and ordered decreasingly according to $>^{rpo}$.

Lemma 6. For all terms $M, N \in \mathcal{B}$ if $M \xrightarrow{loc} N$, then $\underline{M} >^{rpo} \underline{N}$.

Proof. By induction on the definition of $\xrightarrow{loc}$. As there are many possible reductions, we shall present only a few representative cases. First we give one case

where an inner reduction occurs (we shall write rpo for Definition 3).

- $M \equiv \mathsf{Cut}(\langle a\rangle S, (x)T) \xrightarrow{loc} \mathsf{Cut}(\langle a\rangle S', (x)T) \equiv N$
  (1) $S \xrightarrow{loc} S'$ and $\underline{S} >^{rpo} \underline{S'}$ by assumption and induction
  (2) $\underline{M} = \underline{S} \cdot_m \underline{T}$ and $\underline{N} = \underline{S'} \cdot_n \underline{T}$ by Definition 7
  (3) $m \geq n$ by Lemma 5(i)
  (4) $\underline{S} \cdot_m \underline{T} >^{rpo} \underline{S'}$, $\underline{S} \cdot_m \underline{T} >^{rpo} \underline{T}$, $\{\!|\underline{S}, \underline{T}|\!\} >^{rpo}_{mult} \{\!|\underline{S'}, \underline{T}|\!\}$ by (1) and rpo(i)
  (5) $\underline{M} >^{rpo} \underline{N}$ by (4) and rpo(ii,iii)

We now show two typical cases where an $\xrightarrow{x}$-reduction is performed

- $M \equiv \mathsf{C\overleftarrow{ut}}(\langle c\rangle \mathsf{And}_R(\langle a\rangle S, \langle b\rangle T, c), (x)U)$
  $\xrightarrow{x} \mathsf{Cut}(\langle c\rangle \mathsf{And}_R(\langle a\rangle \mathsf{C\overleftarrow{ut}}(\langle c\rangle S, (x)U), \langle b\rangle \mathsf{C\overleftarrow{ut}}(\langle c\rangle T, (x)U), c), (x)U) \equiv N$
  (1) $\underline{M} = (\!|\underline{S}, \underline{T}|\!)\, \langle \underline{U}\rangle_m$ and $\underline{N} = (\!|\underline{S}\langle \underline{U}\rangle_r, \underline{T}\langle \underline{U}\rangle_s|\!) \cdot_t \underline{U}$ by Definition 7
  (2) $m \geq t, r, s$ and $\_\langle\_\rangle_m \gg \_ \cdot_t \_$ by Lemma 5(i,ii) and Definition 5
  (3) $(\!|\underline{S}, \underline{T}|\!)\langle \underline{U}\rangle_m >^{rpo} \underline{S}$, $(\!|\underline{S}, \underline{T}|\!)\langle \underline{U}\rangle_m >^{rpo} \underline{U}$, $\{\!|(\!|\underline{S}, \underline{T}|\!), \underline{U}|\!\} >^{rpo}_{mult} \{\!|\underline{S}, \underline{U}|\!\}$ by rpo(i)
  (4) $(\!|\underline{S}, \underline{T}|\!)\langle \underline{U}\rangle_m >^{rpo} \underline{S}\langle \underline{U}\rangle_r$ by (3) and rpo(ii,iii)
  (5) $(\!|\underline{S}, \underline{T}|\!)\langle \underline{U}\rangle_m >^{rpo} \underline{T}\langle \underline{U}\rangle_s$ analogous to (3,4)
  (6) $(\!|\underline{S}, \underline{T}|\!)\langle \underline{U}\rangle_m >^{rpo} (\!|\underline{S}\langle \underline{U}\rangle_r, \underline{T}\langle \underline{U}\rangle_s|\!)$ by (4,5) and rpo(ii)
  (7) $(\!|\underline{S}, \underline{T}|\!)\langle \underline{U}\rangle_m >^{rpo} \underline{U}$ by rpo(i)
  (8) $\underline{M} >^{rpo} \underline{N}$ by (2,6,7) and rpo(ii)
- $M \equiv \mathsf{C\overleftarrow{ut}}(\langle d\rangle \mathsf{And}_R(\langle a\rangle S, \langle b\rangle T, c), (x)U)$
  $\xrightarrow{x} \mathsf{And}_R(\langle a\rangle \mathsf{C\overleftarrow{ut}}(\langle d\rangle S, (x)U), \langle b\rangle \mathsf{C\overleftarrow{ut}}(\langle d\rangle T, (x)U), c) \equiv N$
  (1) $\underline{M} = (\!|\underline{S}, \underline{T}|\!)\, \langle \underline{U}\rangle_m$ and $\underline{N} = (\!|\underline{S}\langle \underline{U}\rangle_r, \underline{T}\langle \underline{U}\rangle_s|\!)$ by Definition 7
  (2) $m \geq r, s$ by Lemma 5(i,ii)
  (3) $(\!|\underline{S}, \underline{T}|\!)\langle \underline{U}\rangle_m >^{rpo} \underline{S}$, $(\!|\underline{S}, \underline{T}|\!)\langle \underline{U}\rangle_m >^{rpo} \underline{U}$, $\{\!|(\!|\underline{S}, \underline{T}|\!), \underline{U}|\!\} >^{rpo}_{mult} \{\!|\underline{S}, \underline{U}|\!\}$ by rpo(i)
  (4) $(\!|\underline{S}, \underline{T}|\!)\langle \underline{U}\rangle_m >^{rpo} \underline{S}\langle \underline{U}\rangle_r$ by (3) and rpo(ii,iii)
  (5) $(\!|\underline{S}, \underline{T}|\!)\langle \underline{U}\rangle_m >^{rpo} \underline{T}\langle \underline{U}\rangle_s$ analogous to (3,4)
  (6) $\underline{M} >^{rpo} \underline{N}$ by (4,5) and rpo(ii)

Last we tackle two cases, one where a commuting reduction and one where a logical reduction occurs.

- $M \equiv \mathsf{Cut}(\langle a\rangle S, (x)T) \xrightarrow{c} \mathsf{C\overleftarrow{ut}}(\langle a\rangle S, (x)T) \equiv N$
  (1) $\underline{M} = \underline{S} \cdot_m \underline{T}$ and $\underline{N} = \underline{S}\langle \underline{T}\rangle_n$ by Definition 7
  (2) $m > n$ and $\_ \cdot_m \_ \gg \_\langle\_\rangle_n$ by Lemma 5(iii) and Definition 5
  (3) $\underline{S} \cdot_m \underline{T} >^{rpo} \underline{S}$ and $\underline{S} \cdot_m \underline{T} >^{rpo} \underline{T}$ by rpo(i)
  (4) $\underline{M} >^{rpo} \underline{N}$ by (2,3) and rpo(ii)
- $M \equiv \mathsf{Cut}(\langle c\rangle \mathsf{And}_R(\langle a\rangle S, \langle b\rangle T, c), (y)\mathsf{And}^1_L((x)U, y)) \xrightarrow{l} \mathsf{Cut}(\langle a\rangle S, (x)U) \equiv N$
  (1) $\underline{M} = (\!|\underline{S}, \underline{T}|\!) \cdot_m (\!|\underline{U}|\!)$ and $\underline{N} = \underline{S} \cdot_n \underline{U}$ by Definition 7
  (2) $m > n$ by Lemma 5(iii)
  (3) $(\!|\underline{S}, \underline{T}|\!) \cdot_m (\!|\underline{U}|\!) >^{rpo} \underline{S}$, $(\!|\underline{S}, \underline{T}|\!) \cdot_m (\!|\underline{U}|\!) >^{rpo} \underline{U}$ by rpo(i)
  (4) $\underline{M} >^{rpo} \underline{N}$ by (2,3) and rpo(ii)

Using this lemma we can show that every $\xrightarrow{loc}$-reduction sequence starting from a term belonging to $\mathcal{TU}_\wedge$ is terminating.

Lemma 7. Every $\xrightarrow{loc}$-reduction sequence starting with a term that belongs to $\mathcal{TU}_\wedge$ is terminating.

Proof. Suppose for the sake of deriving a contradiction that from M the infinite reduction sequence $M \equiv M_1 \xrightarrow{loc} M_2 \xrightarrow{loc} M_3 \xrightarrow{loc} M_4 \xrightarrow{loc} \ldots$ starts. Because M is completely unlabelled we have for all subterms N of M that $|N|_x \equiv N$, and because M is well-typed we know that each of them is strongly normalising under $\xrightarrow{gbl}$. Consequently, every $\text{MAXRED}_{glb}(|N|_x)$ is finite, and thus M is bounded. By Lemmas 4 and 7 we have that the infinite reduction sequence starting from M can be mapped onto the decreasing chain $\underline{M_1} >^{rpo} \underline{M_2} >^{rpo} \underline{M_3} >^{rpo} \underline{M_4} >^{rpo} \ldots$ which however contradicts the well-foundedness of $>^{rpo}$. Thus all $\xrightarrow{loc}$-reduction sequences starting with a term that is an element in $\mathcal{TU}_\wedge$ must terminate. □

Next, we extend this lemma to all terms of $\mathcal{T}_\wedge$. To do so, we shall first show that for every $M \in \mathcal{T}_\wedge$ there is a term $N \in \mathcal{TU}_\wedge$, such that $N \xrightarrow{loc}{}^* M$. Because N is an element in $\mathcal{TU}_\wedge$, we have that N is strongly normalising by the lemma just given, and so M, too, must be strongly normalising.

Lemma 8. For every term $M \in \mathcal{T}_\wedge$ with the typing judgement $\Gamma \triangleright M \triangleright \Delta$, there is a term $N \in \mathcal{TU}_\wedge$ with the typing judgement $\Gamma', \Gamma \triangleright N \triangleright \Delta, \Delta'$ such that $N \xrightarrow{loc}{}^* M$.

Proof. We construct N by inductively replacing in M all occurrences of $\mathsf{C\overleftarrow{ut}}$ and $\mathsf{C\overrightarrow{ut}}$ by some instances of Cut. We analyse the case where $\mathsf{C\overleftarrow{ut}}(\langle a\rangle S, (x)T)$ is a subterm of M.

- If the subterm S does not freshly introduce a, then we replace $\mathsf{C\overleftarrow{ut}}(\langle a\rangle S, (x)T)$ simply by $\mathsf{Cut}(\langle a\rangle S, (x)T)$ (both terms have the same typing judgement). In this case we have $\mathsf{Cut}(\langle a\rangle S, (x)T) \xrightarrow{c} \mathsf{C\overleftarrow{ut}}(\langle a\rangle S, (x)T)$.
- The more interesting case is where S freshly introduces a. Here we cannot simply replace $\mathsf{C\overleftarrow{ut}}$ with Cut, because there is no reduction with $N \xrightarrow{loc}{}^* M$. Therefore we replace $\mathsf{C\overleftarrow{ut}}(\langle a\rangle S, (x)T)$ by $\mathsf{Cut}(\langle a\rangle \mathsf{Cut}(\langle b\rangle S, (y)\mathsf{Ax}(y,c)), (x)T)$ in which b and c are fresh co-names that do not occur anywhere else (this ensures that the new cut-instances are well-typed). Now we show how the new term can reduce. Because $\mathsf{Cut}(\langle b\rangle S, (y)\mathsf{Ax}(y,c))$ does not freshly introduce a, we can first perform two commuting reductions and subsequently we can remove the labelled cut by a $\xrightarrow{gc}$-reduction, *viz.*

$$\begin{aligned}\mathsf{Cut}(\langle a\rangle \mathsf{Cut}(\langle b\rangle S, (y)\mathsf{Ax}(y,c)), (x)T) &\xrightarrow{c} \mathsf{C\overleftarrow{ut}}(\langle a\rangle \mathsf{Cut}(\langle b\rangle S, (y)\mathsf{Ax}(y,c)), (x)T)\\ &\xrightarrow{c} \mathsf{C\overleftarrow{ut}}(\langle a\rangle \mathsf{C\overleftarrow{ut}}(\langle b\rangle S, (y)\mathsf{Ax}(y,c)), (x)T)\\ &\xrightarrow{gc} \mathsf{C\overleftarrow{ut}}(\langle a\rangle S, (x)T)\end{aligned}$$

□

Now the proof of Theorem 1 is by a simple contradiction argument.

Proof of Theorem 1. Suppose $M \in \mathcal{T}_\wedge$ is not strongly normalising. Then by the lemma just given there is a term $N \in \mathcal{TU}_\wedge$ such that $N \xrightarrow{loc}{}^* M$. Clearly, if M is not strongly normalising, then so is N, which however contradicts Lemma 7. Consequently, M must be strongly normalising. □

6 Conclusion

In this paper we considered the problem of defining a strongly normalising cut-elimination procedure for classical logic that satisfies the three criteria given in the introduction and that is Gentzen-like. While Gentzen-like cut-elimination procedures tend to break strong normalisation, in this paper we have shown that this property can be retained by introducing labelled cuts. For reasons of space we have given our system for only the $\wedge$-fragment. However, our techniques apply to the other connectives and to the first-order quantifiers. This should provide us with a bridge between our earlier calculus [11, 12] and an implementation.

There are many directions for further work. For example what is the precise correspondence in the intuitionistic case between normalisation in the lambda-calculus (with explicit substitutions) and our strongly normalising cut-elimination procedure? This is of interest since the Gentzen-like cut-elimination procedure presented in this paper is rather helpful in proving strong normalisation of other reduction systems by simple translations (e.g. the lambda-calculus, $\lambda\mathtt{x}$ and Parigot's $\lambda\mu$). Some of these issues are addressed in [11].

References

1. R. Bloo and H. Geuvers. Explicit Substitution: On the Edge of Strong Normalisation. *Theoretical Computer Science*, 211(1–2):375–395, 1999.
2. V. Danos, J.-B. Joinet, and H. Schellinx. A New Deconstructive Logic: Linear Logic. *Journal of Symbolic Logic*, 62(3):755–807, 1997.
3. N. Dershowitz. Orderings for Term Rewriting Systems. *Theoretical Computer Science*, 17:279–301, 1982.
4. R. Dyckhoff and L. Pinto. Cut-Elimination and a Permutation-Free Sequent Calculus for Intuitionistic Logic. *Studia Logica*, 60(1):107–118, 1998.
5. J. Gallier. Constructive Logics. Part I: A Tutorial on Proof Systems and Typed λ-calculi. *Theoretical Computer Science*, 110(2):249–239, 1993.
6. G. Gentzen. Untersuchungen über das logische Schließen I and II. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1935.
7. H. Herbelin. A λ-calculus Structure Isomorphic to Sequent Calculus Structure. In *Computer Science Logic*, volume 933 of *LNCS*, pages 67–75. Springer Verlag, 1994.
8. J. M. E. Hyland. Proof Theory in the Abstract. *Annals of Pure and Applied Logic*, 2000. To appear.
9. P. A. Melliès. Typed Lambda Calculi with Explicit Substitutions May Not Terminate. In *Typed Lambda Calculi and Applications*, volume 902 of *LNCS*, pages 328–334. Springer Verlag, 1995.
10. K. H. Rose. Explicit Substitution: Tutorial & Survey. Technical report, BRICS, Department of Computer Science, University of Aarhus, 1996.
11. C. Urban. *Classical Logic and Computation*. PhD thesis, Cambridge University, October 2000.
12. C. Urban and G. M. Bierman. Strong Normalisation of Cut-Elimination in Classical Logic. *Fundamenta Informaticae*, 45(1–2):123–155, 2001.