Fundamenta Informaticae 44 (2000) 1–34

IOS Press

Strong Normalisation of Cut-Elimination in Classical Logic

C. Urban[*]
Institut de Mathématiques de Luminy,
CNRS Marseille,
Marseille, France.
urban@iml.univ-mrs.fr

G. M. Bierman[†]
Computer Laboratory,
University of Cambridge,
Cambridge, UK.
gmb@cl.cam.ac.uk

Abstract. In this paper we present a strongly normalising cut-elimination procedure for classical logic. This procedure adapts Gentzen's standard cut-reductions, but is less restrictive than previous strongly normalising cut-elimination procedures. In comparison, for example, with works by Dragalin and Danos et al., our procedure requires no special annotations on formulae and allows cut-rules to pass over other cut-rules. In order to adapt the notion of symmetric reducibility candidates for proving the strong normalisation property, we introduce a novel term assignment for sequent proofs of classical logic and formalise cut-reductions as term rewriting rules.

Keywords: Classical Logic, Cut-Elimination, Strong Normalisation, Symmetric Reducibility Candidates.

[*] Address for correspondence: Institut de Mathématiques de Luminy, CNRS Marseille, Marseille, France
[†] Address for correspondence: Computer Laboratory, University of Cambridge, Cambridge, UK

1. Introduction

Gentzen showed in his seminal paper [13] that all applications of the cut-rule can be eliminated from proofs in the sequent calculi LK and LJ. He not only proved that all occurrences of this rule 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 cut-rules from a proof immediately, rather replace every instance of the cut-rule with simpler cut-rules, and by iteration one eventually ends up with a cut-free proof, also called a normal form. Since Gentzen's paper many *Hauptsätze* (cut-elimination theorems) have appeared for various sequent calculus formulations. Each of them proves termination of a particular cut-elimination procedure. In this paper we shall introduce a novel cut-elimination procedure for a sequent calculus of classical logic, whose design is motivated by the following three criteria:

1. the cut-elimination procedure should *not* restrict the collection of normal forms reachable from a given proof such 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 cut-rules to pass over other cut-rules.

At the time of writing, we are not aware of any other cut-elimination procedure for a sequent calculus of classical logic that satisfies all three criteria. So in the remainder of this introduction we shall justify these criteria.

Typically, cut-elimination procedures for classical logic are non-deterministic, in the sense that applying different cut-reductions may lead to different normal forms. With respect to our first criterion, most cut-elimination procedures, including Gentzen's original, are thus quite unsatisfactory since they terminate only if a particular strategy for cut-elimination is employed. Common examples being an innermost reduction strategy, or the elimination of the cut with the highest rank. An unpleasant consequence of these strategies is that they restrict heavily the number of normal forms reachable from a given proof. However, the normal forms reachable from a proof play an important rôle, if we wish to extend the proposition-as-types analogy to classical logic. Therefore our first two criteria.

As a first attempt for a strongly normalising cut-elimination procedure one might simply take an unrestricted version of Gentzen's cut-elimination procedure; that is by removing the strategy. Unfortunately, this would, as stated earlier, allow infinite reduction sequences, one of which is illustrated in the following example taken from [9, 12].

Example 1.1. Consider the proof

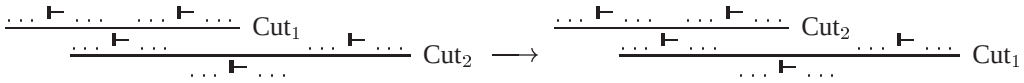
$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A\vee A\vdash A,A}\vee_L}{A\vee A\vdash A}\,\textbf{Contr}_R \qquad \dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A,A\vdash A\wedge A}\wedge_R}{A\vdash A\wedge A}\,\textbf{Contr}_L}{A\vee A\vdash A\wedge A}\,\textbf{Cut}$$

The problem lies with the lower cut-rule—a commuting cut—which needs to be permuted upwards. (A cut-rule is said to be a logical cut when both cut-formulae are introduced by axioms or logical inference rules; otherwise the cut-rule is said to be a commuting cut.) There are two possible cut-reductions: either the cut-rule can be permuted upwards in the left proof branch or in the right proof branch. If one is not

careful, applying these cut-reductions in alternation can lead to arbitrary big normal forms and to non-termination. For example, consider the reduction sequence starting with the proof above and continuing as follows

This proof contains an instance of the cut-reduction applied in the first step (bold face). Even worse, this instance is bigger than in the proof we started with, and so in effect we can construct reduction sequences with possibly infinitely big normal forms. □

It seems difficult to avoid the infinite reduction sequence given in the example above using an unrestricted Gentzen-like formulation of the cut-reductions. A number of people, for example [6, 7, 9, 10, 17], have managed to develop strongly normalising cut-elimination procedures, but they all impose fairly strong restrictions on the cut-reductions. Here is one common restriction: consider the following cut-reduction, which allows a cut-rule (Suffix 2) to pass over another cut-rule (Suffix 1).



Clearly, this cut-reduction would immediately break strong normalisation because the reduct is again an instance of this reduction, and we can loop by constantly applying it. Thus a common restriction is to not allow in any circumstance a cut-rule to pass over another cut-rule. However such a restriction 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 the strong normalisation result of the cut-elimination procedure, as noted in [11, 17]. Therefore our third criterion. We shall design our cut-elimination procedure so that cut-rules can pass over other cut-rules without breaking the strong normalisation property. As a pleasing result, we can simulate beta-reduction and infer strong normalisation of the simply-typed lambda calculus from the strong normalisation result of our cut-elimination procedure. The details of this result appeared in [25].

Danos et al. allow cut-rules to pass over other cut-rules in their strongly normalising cut-elimination procedure given for the sequent calculus LK^{tq} [9, 18]. So this cut-elimination procedure satisfies our second and third criterion, but as we shall see it violates the first. In LK^{tq} every formula (and its subformulae) are required to be coloured with either '$\leftharpoonup$' or '$\rightharpoonup$'. Here is an instance of a cut-rule in LK^{tq}.

$$\dfrac{\overleftarrow{A} \vdash \overleftarrow{\overleftarrow{B} \wedge \overrightarrow{C}} \qquad \overleftarrow{\overleftarrow{B} \wedge \overrightarrow{C}} \vdash \overrightarrow{D}}{\overleftarrow{A} \vdash \overrightarrow{D}}\ \mathrm{Cut}$$

Recall the problematic infinite reduction sequence shown in Example 1.1. This reduction sequence is avoided in their procedure by devising a specific protocol for cut-elimination, which uses the additional information provided by the colours. If in a commuting cut the colour '$\leftharpoonup$' is attached to the cut-formula, then the commuting cut is permuted to the left, and similarly for the '$\rightharpoonup$' colour (hence the use of an arrow to denote a colour!). By enforcing that commuting cuts can be permuted into one direction only, the infinite reduction sequence cannot be constructed.

However, there are two annoying restrictions in their cut-elimination procedure for LK^{tq}, both of which violate our first criterion.

- First, there is a problem with the compositionality of the colour annotation, in the sense that some cut-rules require the same colour annotation for their cut-formulae: the choice of a colouring can

$$\frac{}{x:B,\Gamma \triangleright \mathsf{Ax}(x,a) \triangleright \Delta, a:B}\ \mathrm{Ax} \qquad \frac{\Gamma_1 \triangleright M \triangleright \Delta_1, a:B \qquad x:B, \Gamma_2 \triangleright N \triangleright \Delta_2}{\Gamma_1,\Gamma_2 \triangleright \mathsf{Cut}(\langle a\rangle M, (x)N) \triangleright \Delta_1,\Delta_2}\ \mathrm{Cut}$$

$$\frac{\Gamma \triangleright M \triangleright \Delta, a:B}{x:\neg B,\Gamma \triangleright \mathsf{Not}_L(\langle a\rangle M, x) \triangleright \Delta}\ \neg_L \qquad \frac{x:B,\Gamma \triangleright M \triangleright \Delta}{\Gamma \triangleright \mathsf{Not}_R((x)M, a) \triangleright \Delta, a:\neg B}\ \neg_R$$

$$\frac{x:B_i,\Gamma \triangleright M \triangleright \Delta}{y:B_1 \wedge B_2,\Gamma \triangleright \mathsf{And}^i_L((x)M, y) \triangleright \Delta}\ \wedge_{L_i\,(i=1,2)} \qquad \frac{\Gamma \triangleright M \triangleright \Delta, a:B \qquad \Gamma \triangleright N \triangleright \Delta, b:C}{\Gamma \triangleright \mathsf{And}_R(\langle a\rangle M, \langle b\rangle N, c) \triangleright \Delta, c:B\wedge C}\ \wedge_R$$

$$\frac{x:B,\Gamma \triangleright M \triangleright \Delta \qquad y:C,\Gamma \triangleright N \triangleright \Delta}{z:B\vee C,\Gamma \triangleright \mathsf{Or}_L((x)M,(y)N,z) \triangleright \Delta}\ \vee_L \qquad \frac{\Gamma \triangleright M \triangleright \Delta, a:B_i}{\Gamma \triangleright \mathsf{Or}^i_R(\langle a\rangle M, b) \triangleright \Delta, b:B_1\vee B_2}\ \vee_{R_i\,(i=1,2)}$$

$$\frac{\Gamma \triangleright M \triangleright \Delta, a:B \qquad x:C,\Gamma \triangleright N \triangleright \Delta}{y:B\supset C,\Gamma \triangleright \mathsf{Imp}_L(\langle a\rangle M,(x)N,y) \triangleright \Delta}\ \supset_L \qquad \frac{x:B,\Gamma \triangleright M \triangleright \Delta, a:C}{\Gamma \triangleright \mathsf{Imp}_R((x)\langle a\rangle M, b) \triangleright \Delta, b:B\supset C}\ \supset_R$$

Figure 1. Inference rules for the propositional fragment of classical logic.

permeate through a proof. In particular, the colour annotation has to respect, using terminology introduced for LK^{tq}, *identity classes* [23, Page 107]. For example, when annotating colours to the following LK-proof

$$\dfrac{\dfrac{\vdots}{\vdash B} \qquad \dfrac{\dfrac{}{B \vdash B}}{\vdots\ B,\Gamma \vdash \Delta, B}}{\Gamma \vdash \Delta, B}\ \mathrm{Cut} \qquad \dfrac{\vdots}{B \vdash}}{\Gamma \vdash \Delta}\ \mathrm{Cut}$$

all the occurrences of B must have the same colour. In effect, the normal forms that arise by permuting both cut-rules towards the axiom, where they merge into a single cut-rule, cannot be obtained using the cut-elimination procedure of Danos et al.

- Second, the colour annotation is invariant under cut-reductions. Thus whenever an instance of the cut-rule is duplicated in a reduction sequence, the colour annotation prevents both instances from reducing differently. Figure 2 gives an example of such a reduction sequence that exists in LK, but not in LK^{tq}.

Making the cut-elimination procedure dependent on the colour annotations is, in fact, a very strong restriction: the colours ensure that the cut-elimination procedure becomes confluent; that is deterministic. The confluence result is an essential property in the strong normalisation proof given by Danos et al., because it enabled them to exploit the strong normalisation result for proof nets in linear logic. The colours are used ingeniously to map every LK^{tq}-proof to a corresponding proof-net in linear logic and every cut-elimination step to a series of reductions on proof-nets [14].

The strongly normalising cut-elimination procedure we shall present in this paper includes the standard Gentzen-like cut-reductions for logical cuts. The cut-reductions dealing with commuting cuts, on the other hand, will be simplified versions of the reductions presented by Danos et al. (we remove the

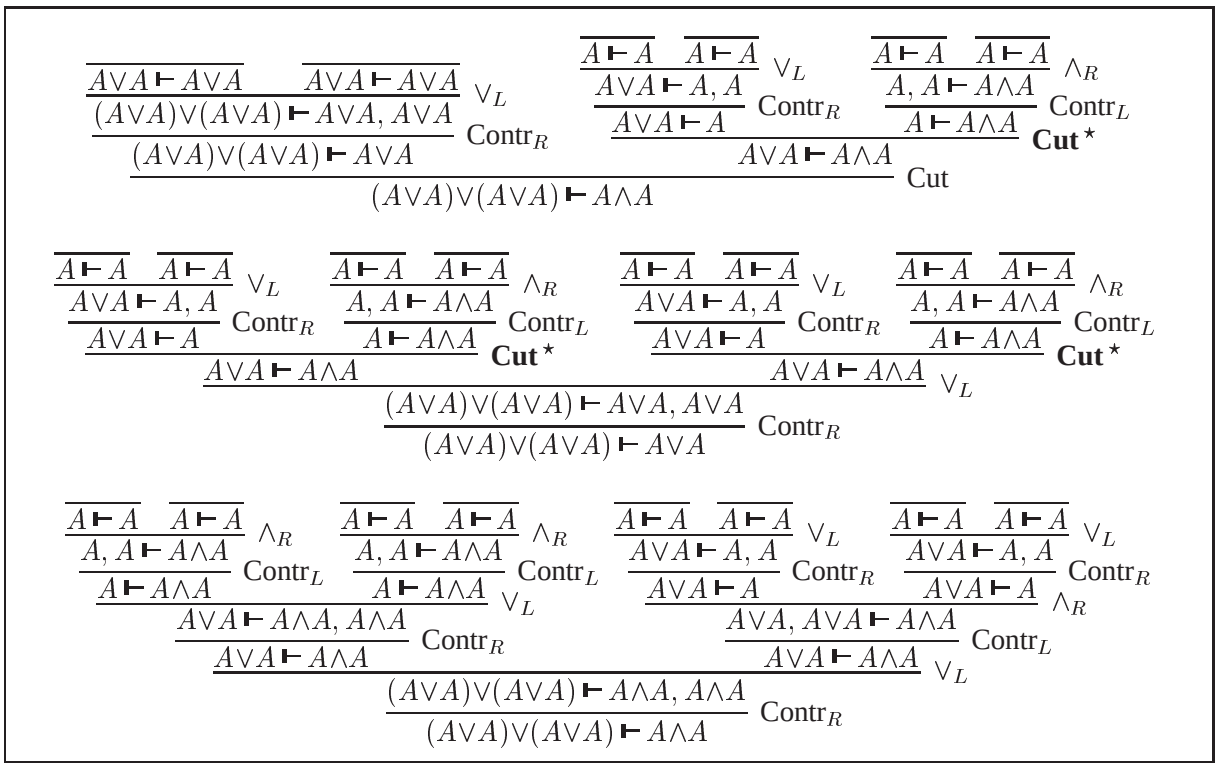


Figure 2. The displayed LK-proofs are taken from a reduction sequence that starts with the first proof and ends with the third proof—a normal form. The second proof is an intermediate step. The cut-rules in the top proof are eliminated in such a way that first the right subproof is duplicated creating two instances of the cut-rule marked with a star (second proof). Subsequently, each copy of this cut-rule is reduced applying different cut-reductions. This reduction sequence is impossible using a cut-elimination procedure that depends on colour annotations, because the colours prevent the two copies of the cut-rule reducing differently. In effect, starting from the first proof the normal form is not reachable in LK^{tq}.

colour annotations). In consequence, we shall show that the colours in LK^{tq} are unnecessary to ensure strong normalisation of cut-elimination. As mentioned earlier, a pleasing consequence is that, in general, more normal forms can be reached from a given proof containing cut-rules. Unfortunately, the generality of our reduction system means that strong normalisation is much more difficult to prove; it cannot, for example, be proved by a translation into proof-nets. In the end, we found it extremely useful to develop a term calculus for sequent derivations. This then allowed us to adapt directly a powerful proof technique from the term rewriting literature.

The paper is organised as follows. In Section 2 we shall introduce a sequent calculus where the inference rules are inspired by Kleene's sequent calculus G3a [19] and the sequent calculus G3c of [24]. One distinguishing feature of our calculus is that the structural rules are completely implicit in the form of the logical rules. In effect, our contexts are sets, as in type-theory, and *not* multisets, as in LK and LJ. We shall annotate the corresponding sequent proofs with terms and formulate the cut-reductions as term rewriting rules. A detailed strong normalisation proof will be given in Section 3. The proof adapts Barbanera and Berardi's technique of symmetric reducibility candidates [2]. In Section 4 we shall give

some details of how to extend the strong normalisation result to the first-order fragment of classical logic. Section 5 will conclude and give suggestions for further work.

2. Terms, Judgements, Rewrite Rules and Substitution

In this section we shall introduce a sequent calculus for classical logic and develop a strongly normalising cut-elimination procedure. In order to present the strong normalisation proof in a convenient form, we shall annotate our sequent proofs with terms and formalise the cut-elimination procedure as a term rewriting system. In particular, we are able to express a proof transformation as a special sort of proof substitution.

Our sequents consist of two contexts—an *antecedent* and a *succedent*—both of which are sets of (label,formula) pairs. 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 names and ... , x, y, z for co-names. Consequently, contexts are either built up by (name,formula) pairs or (co-name,formula) pairs. We shall call the former *left-contexts* and the later *right-contexts*. Furthermore, we shall employ some shorthand notation for contexts: rather than writing, for example, $\{(x, B), (y, C), (z, D)\}$, we shall simply write $x : B, y : C, z : D$ and refer to $\{x, y, z\}$ as the *domain* of this context.

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 a sequent proof, and thus allow us to define a complete cut-elimination procedure as a term rewriting system. Other proposals for terms, for example [3, 21], do not encode the structure of proofs and so would seem less useful for our purposes. The set of raw terms, $\mathcal{R}$, is defined by the grammar

$$
\begin{array}{lcll}
M, N & ::= & \mathsf{Ax}(x, a) & \text{Axiom} \\
 & | & \mathsf{Cut}(\langle a{:}B\rangle M, (x{:}B)N) & \text{Cut} \\
 & | & \mathsf{Not}_R((x{:}B)M, a) & \text{Not-R} \\
 & | & \mathsf{Not}_L(\langle a{:}B\rangle M, x) & \text{Not-L} \\
 & | & \mathsf{And}_R(\langle a{:}B\rangle M, \langle b{:}C\rangle N, c) & \text{And-R} \\
 & | & \mathsf{And}^i_L((x{:}B)M, y) & \text{And-L}_i \quad i = 1, 2 \\
 & | & \mathsf{Or}^i_R(\langle a{:}B\rangle M, b) & \text{Or-R}_i \quad i = 1, 2 \\
 & | & \mathsf{Or}_L((x{:}B)M, (y{:}C)N, z) & \text{Or-L} \\
 & | & \mathsf{Imp}_R((x{:}B)\langle a{:}C\rangle M, b) & \text{Imp-R} \\
 & | & \mathsf{Imp}_L(\langle a{:}B\rangle M, (x{:}C)N, y) & \text{Imp-L}
\end{array}
$$

where x, y, z are taken from a set of names and a, b, c from a set of co-names; B and C are types (formulae) given by the grammar

$$B ::= A \mid \neg B \mid B \wedge B \mid B \vee B \mid B \supset B$$

in which A ranges over propositional symbols.

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 variable can be substituted with a term, 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. For our terms we have the relatively standard notions of free names and free co-names. Given a term, say M, its set of free names and free co-names is written as $FN(M)$ and $FC(M)$, respectively. Another useful notion is as follows.

Definition 2.1. A term, M, *introduces* the name z or co-name c, if and only if M is of the form

for z:	$\mathsf{Ax}(z, c)$	for c:	$\mathsf{Ax}(z, c)$
	$\mathsf{Not}_L(\langle a\rangle S, z)$		$\mathsf{Not}_R((x)S, c)$
	$\mathsf{And}^i_L((x)S, z)$		$\mathsf{And}_R(\langle a\rangle S, \langle b\rangle T, c)$
	$\mathsf{Or}_L((x)S, (y)T, z)$		$\mathsf{Or}^i_R(\langle a\rangle S, c)$
	$\mathsf{Imp}_L(\langle a\rangle S, (x)T, z)$		$\mathsf{Imp}_R((x)\langle a\rangle S, c)$

A term *freshly* introduces a name, if and only if none of its proper subterms introduces this name. In other words, the name must not be free in a proper subterm. Similarly for co-names. □

As we shall see later, this definition corresponds to the traditional notion of the main formula of an inference rule.

Thus sequents, or *typing judgements*, in our sequent calculus are of the form $\Gamma \triangleright M \triangleright \Delta$, where Γ is a left-context, M a term and Δ a right-context. A term, M, is said to be *well-typed*, if $\Gamma \triangleright M \triangleright \Delta$ can be derived given the inference rules shown in Figure 1. For the rest of the paper we shall assume that all terms are well-typed and write $\mathcal{T}$ to denote the set of all well-typed terms. Notice however that there are a number of subtleties concerning contexts implicit in the rules for forming typing judgements. 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 left-context $x : B, x : C$ 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.

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

where the cut-rule is permuted to the left, creating two copies of the right subproof. Now permute the upper cut-rule to the right, which gives the following proof.

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

Second, we have the following conventions for forming contexts in Figure 1: a comma in a conclusion stands for the set union and a comma in a premise stands for the *disjoint* set union. Consider for example the $\supset_R$-rule.

$$\frac{x : B, \Gamma \vdash M \vdash \Delta, a : C}{\Gamma \vdash \mathsf{Imp}_R((x)\langle a\rangle M, b) \vdash \Delta, b : B \supset C} \supset_R$$

This rule introduces the (co-name,formula) pair $b : B \supset C$ in the conclusion, and consequently, b is a free co-name in $\mathsf{Imp}_R((x)\langle a\rangle M, b)$. However, b can already be free in the subterm M, in which case $b : B \supset C$ belongs to Δ. Thus the conclusion of the $\supset_R$-rule is of the form

$$\Gamma \vdash \mathsf{Imp}_R((x)\langle a\rangle M, b) \vdash \Delta \oplus b : B \supset C$$

where $\oplus$ denotes set union. Note that $x : B$ and $a : C$ in the premise are *not* be part of the conclusion because they are intended to become bound. Hence the premise must be of the form

$$x : B \otimes \Gamma \vdash M \vdash \Delta \otimes a : C$$

where $\otimes$ denotes disjoint set union. If the term $\mathsf{Imp}_R((x)\langle a\rangle M, b)$ freshly introduces $b : B \supset C$, then the $\supset_R$-rule is as follows

$$\frac{x : B \otimes \Gamma \vdash M \vdash \Delta \otimes a : C}{\Gamma \vdash \mathsf{Imp}_R((x)\langle a\rangle M, b) \vdash \Delta \otimes b : B \supset C} \supset_R$$

where $b : B \supset C$ is not in Δ.

There is one point worth mentioning in the cut-rule, because it is the only inference rule in our sequent calculus that does not share the contexts, but requires that two contexts are joined on each side of the conclusion. Thus we take this rule to be of the following form.

$$\frac{\Gamma_1 \vdash \Delta_1 \otimes a : B \quad x : B \otimes \Gamma_2 \vdash \Delta_2}{\Gamma_1 \oplus \Gamma_2 \vdash \Delta_1 \oplus \Delta_2} \text{ Cut}$$

In consequence, this rule is only applicable, if it does not break the context convention, which can always be achieved by renaming some labels appropriately. Note 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.

We add now two new syntactic categories of terms. They are *not* proof annotations, but play an important rôle in the definition of proof substitution and in the strong normalisation proof.

Definition 2.2. Let M and N be terms, then $(x{:}B)M$ and $\langle a{:}C\rangle N$ are called *named term* and *co-named term*, respectively. More formally we have the following two families of sets indexed by types:

$$\begin{array}{lcl} \mathcal{T}_{(B)} & \stackrel{\text{def}}{=} & \left\{ (x{:}B)M \;\middle|\; M \in \mathcal{T} \text{ with the typing judgement } x : B, \Gamma \vdash M \vdash \Delta \right\} \\ \mathcal{T}_{\langle C\rangle} & \stackrel{\text{def}}{=} & \left\{ \langle a{:}C\rangle N \;\middle|\; N \in \mathcal{T} \text{ with the typing judgement } \Gamma \vdash N \vdash \Delta, a : C \right\} \end{array}$$

□

Next we focus on the term rewriting rules. One reason for introducing terms is that they greatly simplify the formalisation of the cut-reduction rules, most notably the rules for commuting cuts. In the propositional fragment of the sequent calculus LK there are 508 different cases of commuting cuts! Traditional treatments, for example [12, 13, 15], either omit these cut-reductions entirely or present only a handful of cases. This is unfortunate as a careful study of these reductions sheds some light on the problems of non-termination of cut-elimination. Using the notion of proof substitution, which we shall introduce below, the cut-reductions necessary for dealing with commuting cuts can be formalised in only twenty clauses. Clearly, this is an advantage of our use of terms.

Before we formalise the notion of proof substitution, let us give some intuition behind this operation. Consider the following sequent proof where we have omitted the terms and labels for brevity.

$$\pi_1\left\{\dfrac{\dfrac{\dfrac{}{B\vee C\vdash D\supset E, B\vee C}\,\mathrm{Ax}^\star \quad \dfrac{}{B\vee C, D\vdash E, B\vee C}\,\mathrm{Ax}^\star}{B\vee C\vdash D\supset E, B\vee C}\supset_R}{(B\vee C)\vee(B\vee C)\vdash D\supset E, \boxed{B\vee C}}\vee_L \quad \dfrac{\dfrac{\dfrac{}{B, F\vdash B, C}\,\mathrm{Ax} \quad \dfrac{}{C, F\vdash B, C}\,\mathrm{Ax}}{B\vee C, F\vdash B, C}\vee_L^{\bullet}}{\boxed{B\vee C}, F\wedge G\vdash B, C}\wedge_{L_1}\right\}\pi_2}{(B\vee C)\vee(B\vee C), F\wedge G\vdash D\supset E, B, C}\,\mathrm{Cut}$$

The cut-formula (shaded formula) is neither introduced in the $\supset_R$-rule nor in $\wedge_{L_1}$. Therefore the cut-rule is, by definition, a commuting cut. In π_1 the cut-formula is introduced in the axioms marked with a star, and in π_2, respectively, in the inference rule marked with a disc. Eliminating the cut-rule in the proof above means either to permute the derivation π_2 to the places marked with a star and replace the corresponding axioms with π_2, or to permute π_1 and "cut it against" the inference rule marked with a disc. In the former case the derivation being permuted is duplicated.

We realise these operations at the term level with two symmetric forms of substitution, which we shall write as

$$P[x{:}B := \langle a{:}B\rangle Q] \qquad \text{or} \qquad S[b{:}B := (y{:}B)T]\ .$$

If they are clear from the context, the type annotations in substitutions will be often omitted for brevity.

Returning to our motivating example, assume that M and N are the terms corresponding to the subproofs π_1 and π_2. Thus the terms have the following typing judgements.

$$\begin{array}{c} x : (B\vee C)\vee(B\vee C) \ \triangleright\ M \ \triangleright\ a : D\supset E, b : B\vee C \\ y : B\vee C, z : F\wedge G \ \triangleright\ N \ \triangleright\ c : B, d : C \end{array}$$

Consequently, we have the typing judgement

$$x : (B\vee C)\vee(B\vee C), z : F\wedge G \ \triangleright\ \mathsf{Cut}(\langle b\rangle M, (y)N) \ \triangleright\ a : D\supset E, c : B, d : C$$

for the conclusion of the example proof. The term $M[b := (y)N]$ denotes then the following proof, where we have again omitted all terms and labels for brevity.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{B, F\vdash D\supset E, B, C}\,\mathrm{Ax} \quad \dfrac{}{C, F\vdash D\supset E, B, C}\,\mathrm{Ax}}{B\vee C, F\vdash D\supset E, B, C}\vee_L}{B\vee C, F\wedge G\vdash D\supset E, B, C}\wedge_{L_1} \quad \dfrac{\dfrac{\dfrac{\dfrac{}{B, F, D\vdash E, B, C}\,\mathrm{Ax} \quad \dfrac{}{C, F, D\vdash E, B, C}\,\mathrm{Ax}}{B\vee C, F, D\vdash E, B, C}\vee_L}{B\vee C, F\wedge G, D\vdash E, B, C}\wedge_{L_1}}{B\vee C, F\wedge G\vdash D\supset E, B, C}\supset_R}{(B\vee C)\vee(B\vee C), F\wedge G\vdash D\supset E, B, C}\vee_L$$

Similarly the symmetric case $N[y := \langle b\rangle M]$ denotes the proof

$$\dfrac{\dfrac{\dfrac{\dfrac{}{B\vee C \vdash D\supset E, B\vee C}\,\text{Ax} \quad \dfrac{\dfrac{}{B\vee C, D \vdash E, B\vee C}\,\text{Ax}}{B\vee C \vdash D\supset E, B\vee C}\,\supset_R}{(B\vee C)\vee(B\vee C) \vdash D\supset E, B\vee C}\,\vee_L \quad \dfrac{\dfrac{}{B, F\vdash B, C}\,\text{Ax} \quad \dfrac{}{C, F\vdash B, C}\,\text{Ax}}{B\vee C, F\vdash B, C}\,\vee_L}{(B\vee C)\vee(B\vee C), F \vdash D\supset E, B, C}\,\text{Cut}}{(B\vee C)\vee(B\vee C), E\wedge F \vdash D\supset E, B, C}\,\wedge_{L_1}$$

Before we give the definition of the substitution, it is instructive to look at some further examples. As we have seen, commuting cuts need to permute, or "jump", to the places where the cut-formula is a main formula. At the level of terms this means the cuts need to be permuted to every subterm which introduces the cut-formula. Therefore, whenever a substitution is "next" to a term in which the cut-formula is introduced, the substitution becomes an instance of the Cut-term constructor. In the following two examples we shall write $[\sigma]$ and $[\tau]$ for the substitutions $[c := (x)P]$ and $[x := \langle b\rangle Q]$, respectively.

$$\begin{array}{rcl}
\mathsf{And}_R(\langle a\rangle M, \langle b\rangle N, c)[\sigma] & = & \mathsf{Cut}(\langle c\rangle \mathsf{And}_R(\langle a\rangle\, M[\sigma], \langle b\rangle\, N[\sigma], c), (x)P) \\
\mathsf{Imp}_L(\langle a\rangle M, (y)N, x)[\tau] & = & \mathsf{Cut}(\langle b\rangle Q, (x)\mathsf{Imp}_L(\langle a\rangle\, M[\tau], (y)\, N[\tau], x))
\end{array}$$

In the first term the formula labelled with c is the main formula and in the second the formula labelled with x is the main formula. So in both cases the substitutions "expand" to cuts, and in addition, the substitutions are pushed inside the subterms. This is because there might be several occurrences of c and x: both labels need not have been freshly introduced. An exception applies to axioms, where the substitution is defined differently, as shown below.

$$\begin{array}{rcl}
\mathsf{Ax}(x, a)[x := \langle b\rangle P] & = & P[b \mapsto a] \\
\mathsf{Ax}(x, a)[a := (y)Q] & = & Q[y \mapsto x]
\end{array}$$

Recall that $P[b \mapsto a]$ stands for the term P in which every free occurrence of the co-name b is rewritten to a (similarly $Q[y \mapsto x]$). We are left with the cases where the name or co-name that is being substituted for is not a label of the main formula. In these cases the substitutions are pushed inside the subterms or vanish in case of the axioms. Suppose the substitution $[\sigma]$ is *not* of the form $[z := \ldots]$ and $[a := \ldots]$, then we have the following clauses.

$$\begin{array}{rcl}
\mathsf{Or}_L((x)M, (y)N, z)[\sigma] & = & \mathsf{Or}_L((x)\, M[\sigma], (y)\, N[\sigma], z) \\
\mathsf{Ax}(z, a)[\sigma] & = & \mathsf{Ax}(z, a)
\end{array}$$

Figure 3 gives the complete definition of substitution. We do not need to worry about inserting contraction rules when a term is duplicated, since our contexts are sets of labelled formulae, and thus contractions are made implicitly. Another simplification is due to our use of the Barendregt-style naming convention, because we do not need to worry about possible capture of free names or co-names. Let us now introduce some useful terminology for substitutions.

Terminology 2.1. We shall write $[\sigma]$ to range over substitutions of the form $[x := \langle a\rangle Q]$ and $[b := (y)T]$. In the first case we say x is the domain of $[\sigma]$, written as $dom([\sigma])$, and the co-named term $\langle a\rangle Q$ is the co-domain of $[\sigma]$, written as $codom([\sigma])$. Similarly for the second case. □

1.	$\mathsf{Ax}(x,c)[c := (y)P]$	$\stackrel{\text{def}}{=}$	$P[y \mapsto x]$
2.	$\mathsf{Ax}(y,a)[y := \langle c\rangle P]$	$\stackrel{\text{def}}{=}$	$P[c \mapsto a]$
3.	$\mathsf{Not}_R((x)M,a)[a := (y)P]$	$\stackrel{\text{def}}{=}$	$\mathsf{Cut}(\langle a\rangle \mathsf{Not}_R((x)M[a := (y)P], a), (y)P)$
4.	$\mathsf{Not}_L(\langle a\rangle M,x)[x := \langle c\rangle P]$	$\stackrel{\text{def}}{=}$	$\mathsf{Cut}(\langle c\rangle P, (x)\mathsf{Not}_L(\langle a\rangle M[x := \langle c\rangle P], x))$
5.	$\mathsf{And}_R(\langle a\rangle M, \langle b\rangle N, c)[c := (y)P]$	$\stackrel{\text{def}}{=}$	$\mathsf{Cut}(\langle c\rangle \mathsf{And}_R(\langle a\rangle M[c := (y)P], \langle b\rangle N[c := (y)P], c), (y)P)$
6.	$\mathsf{And}^i_L((x)M,y)[y := \langle c\rangle P]$	$\stackrel{\text{def}}{=}$	$\mathsf{Cut}(\langle c\rangle P, (y)\mathsf{And}^i_L((x)M[y := \langle c\rangle P], y))$
7.	$\mathsf{Or}^i_R(\langle a\rangle M,c)[c := (y)P]$	$\stackrel{\text{def}}{=}$	$\mathsf{Cut}(\langle c\rangle \mathsf{Or}^i_R(\langle a\rangle M[c := (y)P], c), (y)P)$
8.	$\mathsf{Or}_L((x)M,(y)N,z)[z := \langle c\rangle P]$	$\stackrel{\text{def}}{=}$	$\mathsf{Cut}(\langle c\rangle P, (z)\mathsf{Or}_L((x)M[z := \langle c\rangle P], (y)N[z := \langle c\rangle P], z))$
9.	$\mathsf{Imp}_R((x)\langle a\rangle M,b)[b := (y)P]$	$\stackrel{\text{def}}{=}$	$\mathsf{Cut}(\langle b\rangle \mathsf{Imp}_R((x)\langle a\rangle M[b := (y)P], b), (y)P)$
10.	$\mathsf{Imp}_L(\langle a\rangle M,(x)N,y)[y := \langle c\rangle P]$	$\stackrel{\text{def}}{=}$	$\mathsf{Cut}(\langle c\rangle P, (y)\mathsf{Imp}_L(\langle a\rangle M[y := \langle c\rangle P], (x)N[y := \langle c\rangle P], y))$
Otherwise:			
11.	$\mathsf{Ax}(x,a)[\sigma]$	$\stackrel{\text{def}}{=}$	$\mathsf{Ax}(x,a)$
12.	$\mathsf{Cut}(\langle a\rangle M,(x)N)[\sigma]$	$\stackrel{\text{def}}{=}$	$\mathsf{Cut}(\langle a\rangle\ M[\sigma], (x)\ N[\sigma])$
13.	$\mathsf{Not}_R((x)M,a)[\sigma]$	$\stackrel{\text{def}}{=}$	$\mathsf{Not}_R((x)\ M[\sigma], a)$
14.	$\mathsf{Not}_L(\langle a\rangle M,x)[\sigma]$	$\stackrel{\text{def}}{=}$	$\mathsf{Not}_L(\langle a\rangle\ M[\sigma], x)$
15.	$\mathsf{And}_R(\langle a\rangle M, \langle b\rangle N, c)[\sigma]$	$\stackrel{\text{def}}{=}$	$\mathsf{And}_R(\langle a\rangle\ M[\sigma], \langle b\rangle\ N[\sigma], c)$
16.	$\mathsf{And}^i_L((x)M,y)[\sigma]$	$\stackrel{\text{def}}{=}$	$\mathsf{And}^i_L((x)\ M[\sigma], y)$
17.	$\mathsf{Or}^i_R(\langle a\rangle M,b)[\sigma]$	$\stackrel{\text{def}}{=}$	$\mathsf{Or}^i_R(\langle a\rangle\ M[\sigma], b)$
18.	$\mathsf{Or}_L((x)M,(y)N,z)[\sigma]$	$\stackrel{\text{def}}{=}$	$\mathsf{Or}_L((x)\ M[\sigma], (y)\ N[\sigma], z)$
19.	$\mathsf{Imp}_R((x)\langle a\rangle M,b)[\sigma]$	$\stackrel{\text{def}}{=}$	$\mathsf{Imp}_R((x)\langle a\rangle\ M[\sigma], b)$
20.	$\mathsf{Imp}_L(\langle a\rangle M,(x)N,y)[\sigma]$	$\stackrel{\text{def}}{=}$	$\mathsf{Imp}_L(\langle a\rangle\ M[\sigma], (x)\ N[\sigma], y)$

Figure 3. Proof substitution.

Next we focus on the cut-reductions for logical cuts. Consider an instance of an $\wedge_R/\wedge_{L_1}$-cut for which a naïve definition of reduction might be

$$\mathsf{Cut}(\langle c\rangle \mathsf{And}_R(\langle a\rangle M, \langle b\rangle N, c), (y)\mathsf{And}^1_L((x)P, y)) \longrightarrow \mathsf{Cut}(\langle a\rangle M, (x)P)\ .$$

Unfortunately, there is a problem with this reduction rule. In our sequent calculus the structural rules are *implicit*: this means that not only does the calculus have fewer inference rules, but more importantly, we have a very convenient way to define substitution (we do not need explicit contractions when a term is duplicated). On the other hand, there is a subtle side-effect of this design decision. Consider the following instance of the redex above

$$\dfrac{\dfrac{\Gamma_1 \vartriangleright M \vartriangleright \Delta_1, c : B\wedge C, a : B \quad \Gamma_1 \vartriangleright N \vartriangleright \Delta_1, b : C}{\Gamma_1 \vartriangleright \mathsf{And}_R(\langle a\rangle M, \langle b\rangle N, c) \vartriangleright \Delta_1, c : B\wedge C}\wedge_R \quad \dfrac{x : B, \Gamma_2 \vartriangleright P \vartriangleright \Delta_2}{y : B\wedge C, \Gamma_2 \vartriangleright \mathsf{And}^1_L((x)P, y) \vartriangleright \Delta_2}\wedge_{L_1}}{\Gamma_1, \Gamma_2 \vartriangleright \mathsf{Cut}(\langle c\rangle \mathsf{And}_R(\langle a\rangle M, \langle b\rangle N, c), (y)\mathsf{And}^1_L((x)P, y)) \vartriangleright \Delta_1, \Delta_2}\,\mathrm{Cut}$$

where c is a free co-name in M. Our naïve reduction rule would yield

$$\dfrac{\Gamma_1 \vartriangleright M \vartriangleright \Delta_1, c : B\wedge C, a : B \quad x : B, \Gamma_2 \vartriangleright P \vartriangleright \Delta_2}{\Gamma_1, \Gamma_2 \vartriangleright \mathsf{Cut}(\langle a\rangle M, (x)P) \vartriangleright \Delta_1, \Delta_2, c : B\wedge C}\,\mathrm{Cut}\ .$$

Here c has become free in the conclusion! The problem is that the original proof, despite first appearances, is not a logical cut, but in fact a commuting cut, and should really be reduced to

$$\mathsf{And}_R(\langle a\rangle M, \langle b\rangle N, c)[c := (y)\mathsf{And}^1_L((x)P, y)]\ .$$

Consequently, we ensure that logical reduction rules apply only where the cut-formula is *freshly* introduced. Figure 4 gives our cut-reductions for logical cuts, denoted by $\xrightarrow{l}$, and commuting cuts, denoted by $\xrightarrow{c}$. We automatically assume that the reductions are closed under context formation, which is a standard convention in term rewriting. For the cut-reductions there are a few remarks worth pointing out.

Remark 2.3. There are a few subtleties in the fourth reduction rule.

- First, there are two ways to reduce a cut-rule having an implication as the cut-formula. Consider the following cut-instance

$$\dfrac{\dfrac{x : B \vartriangleright M \vartriangleright a : C}{\vartriangleright \mathsf{Imp}_R((x)\langle a\rangle M, b) \vartriangleright b : B\supset C}\supset_R \quad \dfrac{\vartriangleright N \vartriangleright c : B \qquad y : C \vartriangleright P \vartriangleright}{z : B\supset C \vartriangleright \mathsf{Imp}_L(\langle c\rangle N, (y)P, z) \vartriangleright}\supset_L}{\vartriangleright \mathsf{Cut}(\langle b\rangle \mathsf{Imp}_R((x)\langle a\rangle M, b), (z)\mathsf{Imp}_L(\langle c\rangle N, (y)P, z)) \vartriangleright}\,\mathrm{Cut}$$

  which can be reduced to either of the following cut-instances.

$$\dfrac{\dfrac{\dfrac{\vartriangleright N \vartriangleright c : B \quad x : B \vartriangleright M \vartriangleright a : C}{\vartriangleright \mathsf{Cut}(\langle c\rangle N, (x)M) \vartriangleright a : C}\,\mathrm{Cut} \qquad y : C \vartriangleright P \vartriangleright}{\vartriangleright \mathsf{Cut}(\langle a\rangle \mathsf{Cut}(\langle c\rangle N, (x)M), (y)P) \vartriangleright}\,\mathrm{Cut} \qquad \dfrac{\vartriangleright N \vartriangleright c : B \quad \dfrac{x : B \vartriangleright M \vartriangleright a : C \quad y : C \vartriangleright P \vartriangleright}{x : B \vartriangleright \mathsf{Cut}(\langle a\rangle M, (y)P) \vartriangleright}\,\mathrm{Cut}}{\vartriangleright \mathsf{Cut}(\langle c\rangle N, (x)\mathsf{Cut}(\langle a\rangle M, (y)P)) \vartriangleright}\,\mathrm{Cut}$$

  Therefore we have included two reductions, which entails that our cut-elimination procedure is non-deterministic.

Logical Cuts ($i = 1, 2$)

1. $\mathsf{Cut}(\langle a\rangle \mathsf{Not}_R((x)M, a), (y)\mathsf{Not}_L(\langle b\rangle N, y)) \xrightarrow{l} \mathsf{Cut}(\langle b\rangle N, (x)M)$
   if $\mathsf{Not}_R((x)M, a)$ and $\mathsf{Not}_L(\langle b\rangle N, y)$ freshly introduce a and y

2. $\mathsf{Cut}(\langle b\rangle \mathsf{And}_R(\langle a_1\rangle M_1, \langle a_2\rangle M_2, b), (y)\mathsf{And}^i_L((x)N, y)) \xrightarrow{l} \mathsf{Cut}(\langle a_i\rangle M_i, (x)N)$
   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

3. $\mathsf{Cut}(\langle b\rangle \mathsf{Or}^i_R(\langle a\rangle M, b), (y)\mathsf{Or}_L((x_1)N_1, (x_2)N_2, y)) \xrightarrow{l} \mathsf{Cut}(\langle a\rangle M, (x_i)N_i)$
   if $\mathsf{Or}^i_R(\langle a\rangle M, b)$ and $\mathsf{Or}_L((x_1)N_1, (x_2)N_2, y)$ freshly introduce b and y

4. $\mathsf{Cut}(\langle b\rangle \mathsf{Imp}_R((x)\langle a\rangle M, b), (z)\mathsf{Imp}_L(\langle c\rangle N, (y)P, z))$
   $\xrightarrow{l} \mathsf{Cut}(\langle a\rangle \mathsf{Cut}(\langle c\rangle N, (x)M), (y)P)$ or
   $\xrightarrow{l} \mathsf{Cut}(\langle c\rangle N, (x)\mathsf{Cut}(\langle a\rangle M, (y)P))$
   if $\mathsf{Imp}_R((x)\langle a\rangle M, b)$ and $\mathsf{Imp}_L(\langle c\rangle N, (y)P, z)$ freshly introduce b and z

5. $\mathsf{Cut}(\langle a\rangle M, (x)\mathsf{Ax}(x, b)) \xrightarrow{l} M[a \mapsto b]$
   if M freshly introduces a

6. $\mathsf{Cut}(\langle a\rangle \mathsf{Ax}(y, a), (x)M) \xrightarrow{l} M[x \mapsto y]$
   if M freshly introduces x

Commuting Cuts

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

Figure 4. Cut-reductions for logical and commuting cuts.

- Second, special care needs to be taken so that there is no clash between bound and free (co-)names. The term $\mathsf{Imp}_R((x)\langle a\rangle M, b)$ binds x and a simultaneously; however in the reducts the cut-rules bind x and a, separately. Therefore in the first reduction rule we need to ensure that a is not a free co-name in $\langle c\rangle N$ and in the second rule that x is not a free name in $(y)P$. This can always be achieved by renaming a and x appropriately: they are binders in $\mathsf{Imp}_R((x)\langle a\rangle M, b)$. We assume that the renaming is done implicitly in the cut-elimination procedure. □

We are now ready to formulate our cut-elimination procedure. We shall define it in terms of an abstract reduction system [1].

Definition 2.4. (Cut-Elimination Procedure)
The cut-elimination procedure $(\mathcal{T}, \xrightarrow{cut})$ is an abstract reduction system where:

- $\mathcal{T}$ is the set of terms, and
- $\xrightarrow{cut}$ consists of the reductions for logical cuts and commuting cuts, i.e.,

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

□

Notice that $\xrightarrow{l}$ and $\xrightarrow{c}$ are closed under context formation. The *completeness* of $\xrightarrow{cut}$ is simply the fact, obvious by inspection, that every term beginning with a cut matches at least one left-hand side of the reduction rules. So each irreducible term, also called a normal form, is cut-free.

We should like to prove that the cut-reductions satisfy the subject reduction property, which states that a term reduces to a term with the same typing judgement.

Proposition 2.5. (Subject Reduction)
Suppose M is a term with the typing judgement $\Gamma \triangleright M \triangleright \Delta$ and $M \xrightarrow{cut} N$, then N is a term with the typing judgement $\Gamma \triangleright N \triangleright \Delta$.

Proof: By inspection of the reduction rules. □

3. Proof of Strong Normalisation

In this section we shall give the details for the strong normalisation proof of the reduction system $(\mathcal{T}, \xrightarrow{cut})$. The proof adapts the technique of the symmetric reducibility candidates from [2]. Unfortunately, we cannot apply this technique directly to prove strong normalisation for $(\mathcal{T}, \xrightarrow{cut})$, because to strengthen an induction hypothesis we need the property

$$M[x := \langle a\rangle P][b := (y)Q] \;\equiv\; M[b := (y)Q][x := \langle a\rangle P]$$

for b not free in $\langle a\rangle P$ and x not free in $(y)Q$. However, this property does *not* hold for the substitution operation given in Figure 3. This means that "independent" substitutions, in general, do not commute! The (only) problematic case is where M is of the form $\mathsf{Ax}(x, b)$; for example

$$\begin{aligned}\mathsf{Ax}(x,b)[x := \langle a\rangle P][b := (y)Q] &= P[a \mapsto b][b := (y)Q]\,, \text{ but}\\ \mathsf{Ax}(x,b)[b := (y)Q][x := \langle a\rangle P] &= Q[y \mapsto x][x := \langle a\rangle P]\,.\end{aligned}$$

Clearly, there is no reason for the two resultant terms to be equal. To remedy this situation we shall define an auxiliary cut-reduction system, $(\mathcal{T}, \xrightarrow{aux})$, which has a more subtle definition of substitution including two special clauses to handle the problematic example above. Intuitively we should expect

$$\left.\begin{array}{l} \mathsf{Ax}(x,b)[x := \langle a\rangle P][b := (y)Q] \\ \mathsf{Ax}(x,b)[b := (y)Q][x := \langle a\rangle P] \end{array}\right\} \;=\; \mathsf{Cut}(\langle a\rangle P, (y)Q)\,.$$

The auxiliary substitution, written as $M\{a := (x)N\}$ and $N\{x := \langle a\rangle M\}$, is defined as follows.

Definition 3.1. (Auxiliary Substitution)
The auxiliary substitution consists of the clauses

$$\begin{array}{rcl} \mathsf{Ax}(x,c)\{c := (y)P\} & \stackrel{\text{def}}{=} & \mathsf{Cut}(\langle c\rangle \mathsf{Ax}(x,c), (y)P) \\ \mathsf{Ax}(y,a)\{y := \langle c\rangle P\} & \stackrel{\text{def}}{=} & \mathsf{Cut}(\langle c\rangle P, (y)\mathsf{Ax}(y,a)) \\ \mathsf{Cut}(\langle a\rangle \mathsf{Ax}(x,a), (y)M)\{x := \langle b\rangle P\} & \stackrel{\text{def}}{=} & \mathsf{Cut}(\langle b\rangle P, (y)M\{x := \langle b\rangle P\}) \\ \mathsf{Cut}(\langle a\rangle M, (x)\mathsf{Ax}(x,b))\{b := (y)P\} & \stackrel{\text{def}}{=} & \mathsf{Cut}(\langle a\rangle M\{b := (y)P\}, (y)P) \end{array}$$

and the clauses 3-20 shown in Figure 3, except that $[\_]$ is replaced with $\{\_\}$. □

For the auxiliary substitution we apply the same terminology as for substitutions of the form $[\sigma]$ (see Terminology 2.1). Since we changed the substitution operation, we need to adapt the reduction rule for commuting cuts. The modified rule is as follows

$$\begin{array}{lcll} \mathsf{Cut}(\langle a\rangle M, (x)N) & \xrightarrow{c'} & M\{a := (x)N\} & \text{if } M \text{ does not freshly introduce } a\,, \text{ or} \\ & \xrightarrow{c'} & N\{x := \langle a\rangle M\} & \text{if } N \text{ does not freshly introduce } x\,. \end{array}$$

The auxiliary cut-elimination procedure is then

Definition 3.2. (Auxiliary Cut-Elimination Procedure)
The auxiliary cut-elimination procedure $(\mathcal{T}, \xrightarrow{aux})$ is an abstract reduction system where:

- $\mathcal{T}$ is the set of terms, and
- $\xrightarrow{aux}$ consists of the rules for the logical cuts and the modified reduction for commuting cuts, i.e.,

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

□

Given the proof of Proposition 2.5, it is a routine matter to verify that this reduction system, too, satisfies the subject reduction property. Let us now outline how we shall proceed in our strong normalisation proof for $(\mathcal{T}, \xrightarrow{aux})$.

1. Define the sets of candidates over types using a fixed point construction (Definition 3.4).
2. Prove that candidates are closed under reduction (Lemma 3.12).
3. Show that a named or co-named term in a candidate implies strong normalisation for the corresponding term (Lemma 3.13).

4. Extend the notion of safe substitution to simultaneous substitutions (Definition 3.15).
5. Prove that all terms are strongly normalising (Theorem 3.19).

Finally, we shall show that every $\xrightarrow{cut}$-reduction maps onto a series of $\xrightarrow{aux}$-reductions and thus prove that $(\mathcal{T}, \xrightarrow{cut})$ is strongly normalising, too.

First, we define for every type two candidates, written as $\langle B\rangle$ and (B). These candidates are subsets of named or co-named terms, i.e., $\langle B\rangle \subseteq \mathcal{T}_{\langle B\rangle}$ and $(B) \subseteq \mathcal{T}_{(B)}$. Whilst traditional notions of candidates are defined by a simple induction over types, our candidates are inductively defined over types, but also include fixed point operations. Before we give the definition of the candidates we shall introduce some set operators, which fix certain closure properties for the candidates. Each of the operators is defined over sets of (co-)named terms having a specific term constructor at the top-level.

Definition 3.3. (Set Operators)

$$\text{AXIOMS}_{(B)} \stackrel{\text{def}}{=} \left\{ (x{:}B)\mathsf{Ax}(y,b) \;\middle|\; (x{:}B)\mathsf{Ax}(y,b) \in \mathcal{T}_{(B)} \right\}$$
$$\text{AXIOMS}_{\langle B\rangle} \stackrel{\text{def}}{=} \left\{ \langle a{:}B\rangle\mathsf{Ax}(y,b) \;\middle|\; \langle a{:}B\rangle\mathsf{Ax}(y,b) \in \mathcal{T}_{\langle B\rangle} \right\}$$

Note that x can be equal to y, and a to b. Figure 5 gives the set operators that correspond to the other term constructors. Additionally we have

$$\text{BINDING}_{(B)}(X) \stackrel{\text{def}}{=} \left\{ (x{:}B)M \;\middle|\; \text{for all } \langle a{:}B\rangle P \in X \,.\, M\{x := \langle a{:}B\rangle P\} \in SN \right\}$$
$$\text{BINDING}_{\langle B\rangle}(Y) \stackrel{\text{def}}{=} \left\{ \langle a{:}B\rangle M \;\middle|\; \text{for all } (x{:}B)P \in Y \,.\, M\{a := (x{:}B)P\} \in SN \right\}$$

where we use the notation $T \in SN$ to indicate that T is strongly normalising (relative to $\xrightarrow{aux}$). □

The set operators given in Figure 5 correspond to the properties we need to prove for showing that a logical cut is strongly normalising, and BINDING is sufficient to prove strong normalisation for a commuting cut. In the definition of the candidates we use fixed points of increasing set operators. A set operator, op, is said to be:

$$\begin{array}{lll} \text{increasing,} & \text{if and only if} & S \subseteq S' \;\Rightarrow\; op(S) \subseteq op(S'), \text{ and} \\ \text{decreasing,} & \text{if and only if} & S \subseteq S' \;\Rightarrow\; op(S) \supseteq op(S'). \end{array}$$

We are now ready to define the set operator NEG and the candidates.

Definition 3.4. (Candidates)
The mutually recursive definition over types for NEG and the candidates is as follows.

NEG$_{\langle B\rangle}$:

$$\left.\begin{array}{ll} \text{NEG}_{\langle A\rangle}(X) & \stackrel{\text{def}}{=} \\ \text{NEG}_{\langle \neg C\rangle}(X) & \stackrel{\text{def}}{=} \\ \text{NEG}_{\langle C\wedge D\rangle}(X) & \stackrel{\text{def}}{=} \\ \text{NEG}_{\langle C_1\vee C_2\rangle}(X) & \stackrel{\text{def}}{=} \\ \text{NEG}_{\langle C\supset D\rangle}(X) & \stackrel{\text{def}}{=} \end{array}\right\} \text{AXIOMS}_{\langle B\rangle} \;\cup\; \text{BINDING}_{\langle B\rangle}(X) \;\cup\; \left\{ \begin{array}{l} - \\ \text{NOTRIGHT}_{\langle \neg C\rangle}((C)) \\ \text{ANDRIGHT}_{\langle C\wedge D\rangle}(\langle C\rangle, \langle D\rangle) \\ \bigcup_{i=1,2} \text{ORRIGHT}^{i}_{\langle C_1\vee C_2\rangle}(\langle C_i\rangle) \\ \text{IMPRIGHT}_{\langle C\supset D\rangle}((C), \langle D\rangle) \end{array}\right.$$

$$\textsc{notright}_{\langle \neg B\rangle}(X) \stackrel{\text{def}}{=} \left\{ \begin{array}{l} \langle a{:}\neg B\rangle \mathsf{Not}_R((x{:}B)M, a) \mid \\ \quad \mathsf{Not}_R((x{:}B)M, a) \text{ freshly introduces } a, \\ \quad (x{:}B)M \in X \end{array} \right\}$$

$$\textsc{notleft}_{(\neg B)}(X) \stackrel{\text{def}}{=} \left\{ \begin{array}{l} (x{:}\neg B)\mathsf{Not}_L(\langle a{:}B\rangle M, x) \mid \\ \quad \mathsf{Not}_L(\langle a{:}B\rangle M, x) \text{ freshly introduces } x, \\ \quad \langle a{:}B\rangle M \in X \end{array} \right\}$$

$$\textsc{andright}_{\langle B\wedge C\rangle}(X,Y) \stackrel{\text{def}}{=} \left\{ \begin{array}{l} \langle c{:}B\wedge C\rangle \mathsf{And}_R(\langle a{:}B\rangle M, \langle b{:}C\rangle N, c) \mid \\ \quad \mathsf{And}_R(\langle a{:}B\rangle M, \langle b{:}C\rangle N, c) \text{ freshly introduces } c, \\ \quad \langle a{:}B\rangle M \in X, \\ \quad \langle b{:}C\rangle N \in Y \end{array} \right\}$$

$$\textsc{andleft}^i_{(B_1\wedge B_2)}(X) \stackrel{\text{def}}{=} \left\{ \begin{array}{l} (y{:}B_1\wedge B_2)\mathsf{And}^i_L((x{:}B_i)M, y) \mid \\ \quad \mathsf{And}^i_L((x{:}B_i)M, y) \text{ freshly introduces } y, \\ \quad (x{:}B_i)M \in X \end{array} \right\}$$

$$\textsc{orright}^i_{\langle B_1\vee B_2\rangle}(X) \stackrel{\text{def}}{=} \left\{ \begin{array}{l} \langle b{:}B_1\vee B_2\rangle \mathsf{Or}^i_R(\langle a{:}B_i\rangle M, b) \mid \\ \quad \mathsf{Or}^i_R(\langle a{:}B_i\rangle M, b) \text{ freshly introduces } b, \\ \quad \langle a{:}B_i\rangle M \in X \end{array} \right\}$$

$$\textsc{orleft}_{(B\vee C)}(X,Y) \stackrel{\text{def}}{=} \left\{ \begin{array}{l} (z{:}B\vee C)\mathsf{Or}_L((x{:}B)M, (y{:}C)N, z) \mid \\ \quad \mathsf{Or}_L((x{:}B)M, (y{:}C)N, z) \text{ freshly introduces } z, \\ \quad (x{:}B)M \in X, \\ \quad (y{:}C)N \in Y \end{array} \right\}$$

$$\textsc{impright}_{\langle B\supset C\rangle}(X,Y) \stackrel{\text{def}}{=} \left\{ \begin{array}{l} \langle b{:}B\supset C\rangle \mathsf{Imp}_R((x{:}B)\langle a{:}C\rangle M, b) \mid \\ \quad \mathsf{Imp}_R((x{:}B)\langle a{:}C\rangle M, b) \text{ freshly introduces } b, \\ \quad \text{for all } (z{:}C)P \in Y \;. \\ \qquad\qquad (x{:}B)M\{a := (z)P\} \in X, \\ \quad \text{for all } \langle c{:}B\rangle Q \in X \;. \\ \qquad\qquad \langle a{:}C\rangle M\{x := \langle c\rangle Q\} \in Y \end{array} \right\}$$

$$\textsc{impleft}_{(B\supset C)}(X,Y) \stackrel{\text{def}}{=} \left\{ \begin{array}{l} (y{:}B\supset C)\mathsf{Imp}_L(\langle a{:}B\rangle M, (x{:}C)N, y) \mid \\ \quad \mathsf{Imp}_L(\langle a{:}B\rangle M, (x{:}C)N, y) \text{ freshly introduces } y, \\ \quad \langle a{:}B\rangle M \in X, \\ \quad (x{:}C)N \in Y \end{array} \right\}$$

Figure 5. Definition of the set operators for the propositional connectives.

$\text{NEG}_{(B)}$:

$$\left.\begin{array}{ll}
\text{NEG}_{(A)}(X) & \stackrel{\text{def}}{=} \\
\text{NEG}_{(\neg C)}(X) & \stackrel{\text{def}}{=} \\
\text{NEG}_{(C_1 \wedge C_2)}(X) & \stackrel{\text{def}}{=} \\
\text{NEG}_{(C \vee D)}(X) & \stackrel{\text{def}}{=} \\
\text{NEG}_{(C \supset D)}(X) & \stackrel{\text{def}}{=}
\end{array}\right\} \text{AXIOMS}_{(B)} \cup \text{BINDING}_{(B)}(X) \cup \left\{\begin{array}{l}
- \\
\text{NOTLEFT}_{(\neg C)}(\langle C\rangle) \\
\bigcup_{i=1,2} \text{ANDLEFT}^{i}_{(C_1 \wedge c_2)}((C_i)) \\
\text{ORLEFT}_{(C \vee D)}((C),(D)) \\
\text{IMPLEFT}_{(C \supset D)}(\langle C\rangle,(D))
\end{array}\right.$$

candidates:

$$\begin{array}{lll}
(B) & \stackrel{\text{def}}{=} & X_0 \\
\langle B\rangle & \stackrel{\text{def}}{=} & \text{NEG}_{\langle B\rangle}((B))
\end{array}$$

where X_0 is the least fixed point of the operator $\text{NEG}_{(B)} \circ \text{NEG}_{\langle B\rangle}$. □

Remark 3.5. The least fixed point of the operator $\text{NEG}_{(B)} \circ \text{NEG}_{\langle B\rangle}$ is defined since both $\text{BINDING}_{\langle B\rangle}$ and $\text{BINDING}_{(B)}$ are decreasing operators. Consequently, $\text{NEG}_{\langle B\rangle}$ and $\text{NEG}_{(B)}$ are decreasing. But then $\text{NEG}_{(B)} \circ \text{NEG}_{\langle B\rangle}$ must be increasing, and the least fixed point X_0 exists according to Tarski's fixed point theorem. □

Two basic properties of the candidates are as follows.

Proposition 3.6.

(i) $(B) = \text{NEG}_{(B)}(\langle B\rangle)$ and $\langle B\rangle = \text{NEG}_{\langle B\rangle}((B))$

(ii) $\text{AXIOMS}_{(B)} \subseteq (B)$ and $\text{AXIOMS}_{\langle B\rangle} \subseteq \langle B\rangle$

Proof: (i) follows from Definition 3.4, and (ii) holds trivially since NEG is closed under AXIOMS. □

Let us analyse some of the motivations behind the completely symmetric definition of the candidates.

Remark 3.7. Given the symmetry stated in Lemma 3.6(i), we have a simple method to check whether a named or co-named term belongs to a candidate. For example, take a co-named term of the form $\langle a{:}B \wedge C\rangle M$ for which we wish to know whether it belongs to the candidate $\langle B \wedge C\rangle$. Because of the equation $\langle B \wedge C\rangle = \text{NEG}_{\langle B \wedge C\rangle}((B \wedge C))$ it is sufficient to show that $\langle a{:}B \wedge C\rangle M$ is an element in $\text{NEG}_{\langle B \wedge C\rangle}((B \wedge C))$. By definition of $\text{NEG}_{\langle B \wedge C\rangle}$ we therefore have to show that $\langle a{:}B \wedge C\rangle M$ belongs to at least one of the following three sets:

$$\begin{array}{ll}
\text{(i)} & \text{AXIOMS}_{\langle B \wedge C\rangle} \\
\text{(ii)} & \text{ANDRIGHT}_{\langle B \wedge C\rangle}(\langle B\rangle,\langle C\rangle) \\
\text{(iii)} & \text{BINDING}_{\langle B \wedge C\rangle}((B \wedge C))
\end{array}$$

This means that $\langle a{:}B \wedge C\rangle M$ must satisfy certain conditions depending on its top-level term constructor. For example, in (i) it is required that $\langle a{:}B \wedge C\rangle M$ is of the form $\langle a{:}B \wedge C\rangle \mathsf{Ax}(x,b)$; in (ii) of the form $\langle a{:}B \wedge C\rangle \mathsf{And}_R(\langle b\rangle S, \langle c\rangle T, a)$ and it is presupposed that $\langle b\rangle S$ and $\langle c\rangle T$ belong to $\langle B\rangle$ and to $\langle C\rangle$, respectively; in (iii) it is required that M is strongly normalising under any substitution on a with a named term belonging to the candidate $(B \wedge C)$. □

In the next four lemmas we deduce some properties of $\{\sigma\}$ and $\xrightarrow{aux}$. The first shows that the standard substitution lemma holds for the auxiliary substitution (this lemma fails for substitutions of the form $[\sigma]$).

Lemma 3.8. (Substitution Lemma)
For all $M \in \mathcal{T}$ and two arbitrary proof substitutions, $\{\sigma\}$ and $\{\tau\}$, such that $dom(\{\sigma\})$ is not free in $codom(\{\tau\})$, we have $M\{\sigma\}\{\tau\} \equiv M\{\tau\}\{\sigma\{\tau\}\}$.

Proof: By induction on the structure of M. The only case that is non-trivial is where M is an axiom, the details of which are given below.

Case $M \equiv \mathsf{Ax}(x, b)$**:** Suppose $\{\sigma\}$ and $\{\tau\}$ are of the form $\{x := \langle a\rangle P\}$ and $\{b := (y)Q\}$, respectively. We analyse in turn the cases $\mathsf{Ax}(x, b)\{\sigma\}\{\tau\}$ and $\mathsf{Ax}(x, b)\{\tau\}\{\sigma\{\tau\}\}$.

$$\begin{aligned}
M\{\sigma\}\{\tau\} &= \mathsf{Cut}(\langle a\rangle P, (x)\mathsf{Ax}(x, b))\{b := (y)Q\}\\
&= \mathsf{Cut}(\langle a\rangle P\{b := (y)Q\}, (y)Q)\\
M\{\tau\}\{\sigma\{\tau\}\} &= \mathsf{Cut}(\langle b\rangle \mathsf{Ax}(x, b), (y)Q)\{x := \langle a\rangle P\{b := (y)Q\}\}\\
&= \mathsf{Cut}(\langle a\rangle P\{b := (y)Q\}, (y)Q\{x := \langle a\rangle P\{b := (y)Q\}\})\\
&\equiv \mathsf{Cut}(\langle a\rangle P\{b := (y)Q\}, (y)Q) \qquad \text{because by assumption } x \notin FN((y)Q)
\end{aligned}$$

□

Lemma 3.9. Suppose $M \in \mathcal{T}$ and $M \xrightarrow{aux} M'$.

(i) If M freshly introduces the name x, then M' freshly introduces x.

(ii) If M freshly introduces the co-name a, then M' freshly introduces a.

Proof: If M freshly introduces a name or a co-name, then M cannot be of the form $\mathsf{Cut}(\_, \_)$ (see Definition 2.1). The lemma follows by inspection of the reduction rules of $\xrightarrow{aux}$. □

Lemma 3.10. For all terms $M \in \mathcal{T}$ we have

(i) $M\{x := \langle a\rangle \mathsf{Ax}(y, a)\} \xrightarrow{aux}{}^{*} M[x \mapsto y]$

(ii) $M\{a := (x)\mathsf{Ax}(x, b)\} \xrightarrow{aux}{}^{*} M[a \mapsto b]$

Proof: By routine induction on the structure of M. □

Notation 3.1. The expression $M \xrightarrow{aux}{}^{0/1} M'$ stands for either $M \equiv M'$ or $M \xrightarrow{aux} M'$. □

Lemma 3.11. For an arbitrary substitution $\{\sigma\}$, if $M \xrightarrow{aux} M'$, then $M\{\sigma\} \xrightarrow{aux}{}^{0/1} M'\{\sigma\}$.

Proof: By induction on the structure of M. We illustrate the proof with one case where $M\{\sigma\} \equiv M'\{\sigma\}$ is possible.

Case $M \equiv \mathsf{Cut}(\langle a\rangle \mathsf{Ax}(y,a), (x)S)$**:** Suppose S freshly introduces x and assume $\{\sigma\}$ is of the form $\{y := \langle c\rangle T\}$. Furthermore, let $M \xrightarrow{aux} M'$ with $M' \equiv S[x \mapsto y]$. In the following calculation the equivalence ('0'-case) occurs if $S\{y := \langle c\rangle T\}$ freshly introduces x.

$$\begin{array}{rcl} M\{\sigma\} & \equiv & \mathsf{Cut}(\langle a\rangle \mathsf{Ax}(y,a), (x)S)\{y := \langle c\rangle T\} \\ & = & \mathsf{Cut}(\langle c\rangle T, (x)S\{y := \langle c\rangle T\}) \\ & \xrightarrow[(*)]{aux}{}^{0/1} & S\{y := \langle c\rangle T\}\{x := \langle c\rangle T\} \\ & \equiv & S[x \mapsto y]\{y := \langle c\rangle T\} \equiv M'\{\sigma\} \end{array}$$

$^{(*)}$ because by Barendregt-style naming convention $x \notin FN(\langle c\rangle T)$ □

The next two lemmas establish important properties of the candidates. The first shows that the candidates are closed under reductions, and the second shows how the candidates are linked to the property of strong normalisation.

Lemma 3.12.

(i) If $\langle a{:}B\rangle M \in \langle B\rangle$ and $M \xrightarrow{aux} M'$, then $\langle a{:}B\rangle M' \in \langle B\rangle$.

(ii) If $(x{:}B)M \in (B)$ and $M \xrightarrow{aux} M'$, then $(x{:}B)M' \in (B)$.

Proof: We prove both cases simultaneously by induction on the degree of B (defined as usual). By Proposition 3.6(i) we need to analyse all possible sets where $\langle a{:}B\rangle M$ could be member in. Four representative cases for (i) are given below; the arguments for (ii) are similar and omitted.

Case $\text{AXIOMS}_{\langle B\rangle}$: $\langle a{:}B\rangle M$ cannot be in $\text{AXIOMS}_{\langle B\rangle}$ because axioms do not reduce.

Case $\text{BINDING}_{\langle B\rangle}((B))$:

(1)	$\langle a{:}B\rangle M \in \text{BINDING}_{\langle B\rangle}((B))$	by assumption
(2)	$M\{a := (x{:}B)P\} \in SN$ for all $(x{:}B)P \in (B)$	by Definition 3.3
(3)	$M \xrightarrow{aux} M'$	by assumption
(4)	$M\{a := (x{:}B)P\} \xrightarrow{aux}{}^{0/1} M'\{a := (x{:}B)P\}$	by Lemma 3.11
(5)	$M'\{a := (x{:}B)P\} \in SN$ for all $(x{:}B)P \in (B)$	by (2) and (4)
(6)	$\langle a{:}B\rangle M' \in \text{BINDING}_{\langle B\rangle}((B))$	by Definition 3.3
(7)	$\langle a{:}B\rangle M' \in \langle B\rangle$	by Definition 3.4

Case $\text{ANDRIGHT}_{\langle C\wedge D\rangle}(\langle C\rangle, \langle D\rangle)$, $B \equiv C\wedge D$:

(1)	$M \equiv \mathsf{And}_R(\langle d\rangle S, \langle e\rangle T, a)$, $M' \equiv \mathsf{And}_R(\langle d\rangle S', \langle e\rangle T', a)$ and $\langle a{:}C\wedge D\rangle M \in \text{ANDRIGHT}_{\langle C\wedge D\rangle}(\langle C\rangle, \langle D\rangle)$	by assumption
(2)	$S \xrightarrow{aux} S'$ and $T \equiv T'$ (the other case being similar)	new assumption
(3)	M freshly introduces a, $\langle d{:}C\rangle S \in \langle C\rangle$ and $\langle e{:}D\rangle T \in \langle D\rangle$	by Definition 3.3
(4)	M' freshly introduces a	by Lemma 3.9
(5)	$\langle d{:}C\rangle S' \in \langle C\rangle$	by induction
(6)	$\langle a{:}C\wedge D\rangle M' \in \text{ANDRIGHT}_{\langle C\wedge D\rangle}(\langle C\rangle, \langle D\rangle)$	by (4), (5) and Definition 3.3
(7)	$\langle a{:}C\wedge D\rangle M' \in \langle C\wedge D\rangle$	by Definition 3.4

Case $\text{IMPRIGHT}_{\langle C\supset D\rangle}((C),\langle D\rangle)$, $B \equiv C\supset D$:

(1) $M \equiv \mathsf{Imp}_R((x)\langle d\rangle S, a)$, $M' \equiv \mathsf{Imp}_R((x)\langle d\rangle S', a)$, $S \xrightarrow{aux} S'$ and $\langle a{:}C\supset D\rangle M \in \text{IMPRIGHT}_{\langle C\supset C\rangle}((C),\langle D\rangle)$ by assumption

(2) M freshly introduces a, $\langle d{:}D\rangle S\{x := \langle e{:}C\rangle Q\} \in \langle D\rangle$ for all $\langle e{:}C\rangle Q \in \langle C\rangle$, and $(x{:}C)S\{d := (z{:}D)P\} \in (C)$ for all $(z{:}D)P \in (D)$ by Definition 3.3

(3) M' freshly introduces a by Lemma 3.9

(4) $S\{x := \langle e{:}C\rangle Q\} \xrightarrow{aux}{}^{0/1} S'\{x := \langle e{:}C\rangle Q\}$, $S\{d := (z{:}D)P\} \xrightarrow{aux}{}^{0/1} S'\{d := (z{:}D)P\}$ by Lemma 3.11

(5) $\langle d{:}D\rangle S'\{x := \langle e{:}C\rangle Q\} \in \langle D\rangle$ for all $\langle e{:}C\rangle Q \in \langle C\rangle$, $(x{:}C)S'\{d := (z{:}D)P\} \in (C)$ for all $(z{:}D)P \in (D)$ by (2) and (4): '0'-case trivial, '1'-case by induction

(6) $\langle a{:}C\supset D\rangle M' \in \text{IMPRIGHT}_{\langle C\supset C\rangle}((C),\langle D\rangle)$ by (3), (5) and Definition 3.3

(7) $\langle a{:}C\supset D\rangle M' \in \langle C\supset D\rangle$ by Definition 3.4

□

Lemma 3.13.

(i) If $\langle a{:}B\rangle M \in \langle B\rangle$, then $M \in SN$.

(ii) If $(x{:}B)M \in (B)$, then $M \in SN$.

Proof: The proof is similar to the one of Lemma 3.12. We shall give the details for four cases of (i).

Case $\text{AXIOMS}_{\langle B\rangle}$: In this case M is an axiom, and therefore strongly normalising.

Case $\text{BINDING}_{\langle B\rangle}((B))$:

(1) $\langle a{:}B\rangle M \in \text{BINDING}_{\langle B\rangle}((B))$ by assumption

(2) $M\{a := (x{:}B)P\} \in SN$ for all $(x{:}B)P \in (B)$ by Definition 3.3

(3) $(x{:}B)\mathsf{Ax}(x,a) \in (B)$ by Lemma 3.6(ii)

(4) $M\{a := (x{:}B)\mathsf{Ax}(x,a)\} \in SN$ by (2), (3) and $P \equiv \mathsf{Ax}(x,a)$

(5) $M\{a := (x{:}B)\mathsf{Ax}(x,a)\} \xrightarrow{aux}{}^{*} M$ by Lemma 3.10

(6) $M \in SN$ by (4) and (5)

Case $\text{ANDRIGHT}_{\langle C\wedge D\rangle}(\langle C\rangle,\langle D\rangle)$, $B \equiv C\wedge D$:

(1) $M \equiv \mathsf{And}_R(\langle d\rangle S, \langle e\rangle T, a)$, $\langle a{:}C\wedge D\rangle M \in \text{ANDRIGHT}_{\langle C\wedge D\rangle}(\langle C\rangle,\langle D\rangle)$ by assumption

(2) $\langle d{:}C\rangle S \in \langle C\rangle$ and $\langle e{:}D\rangle T \in \langle D\rangle$ by Definition 3.3

(3) $S \in SN$ and $T \in SN$ by induction

(4) $\mathsf{And}_R(\langle d\rangle S, \langle e\rangle T, a) \in SN$ by (3)

Case $\text{IMPRIGHT}_{\langle C\supset D\rangle}((C),\langle D\rangle)$, $B \equiv C\supset D$:

(1) $M \equiv \mathsf{Imp}_R((x)\langle d\rangle S, a),\ \langle a{:}C \supset D\rangle M \in \text{IMPRIGHT}_{\langle C\supset C\rangle}((C), \langle D\rangle)$ by assumption
(2) $(x{:}C)S\{d := (z{:}D)P\} \in (C)$ for all $(z{:}D)P \in (D)$ by Definition 3.3
(3) $(z{:}D)\mathsf{Ax}(z, d) \in (D)$ by Lemma 3.6(ii)
(4) $(x{:}C)S\{d := (z{:}D)\mathsf{Ax}(z, d)\} \in (C)$ by (2), (3) and $P \equiv \mathsf{Ax}(z, d)$
(5) $S\{d := (z{:}D)\mathsf{Ax}(z, d)\} \in SN$ by induction
(6) $S\{d := (z{:}D)\mathsf{Ax}(z, d)\} \xrightarrow{aux}{}^{*} S$ by Lemma 3.10
(7) $S \in SN$ by (5) and (6)
(8) $\mathsf{Imp}_R((x)\langle d\rangle S, a) \in SN$ by (7)
□

We are now in the position to prove that a cut is strongly normalising given that its immediate subterms are strongly normalising and in a candidate corresponding to the cut-formula. The proof of this lemma is inspired by a technique applied in [22]. Unfortunately, this proof is rather lengthy: the cases for the logical reductions require relatively difficult arguments.

Lemma 3.14.
If $M, N \in SN$ and $\langle a{:}B\rangle M \in \langle B\rangle$, $(x{:}B)N \in (B)$, then $\mathsf{Cut}(\langle a{:}B\rangle M, (x{:}B)N) \in SN$.

Proof: We prove by induction that all terms to which $\mathsf{Cut}(\langle a\rangle M, (x)N)$ reduces in one step are strongly normalising. The induction proceeds over a lexicographically ordered induction value of the form $(\delta, l(M), l(N))$, where δ is the degree of the cut-formula B; $l(M)$ and $l(N)$ are the lengths of the maximal reduction sequences starting from M and N, respectively. By assumption both $l(M)$ and $l(N)$ are finite.

Inner Reduction:

(1) $\mathsf{Cut}(\langle a\rangle M, (x)N) \xrightarrow{aux} \mathsf{Cut}(\langle a\rangle M', (x)N')$, $\langle a{:}B\rangle M \in \langle B\rangle$ and $(x{:}B)N \in (B)$ by assumption
(2) $M \xrightarrow{aux} M'$ and $N \equiv N'$ (the other case being similar) new assumption
(3) $\langle a{:}B\rangle M' \in \langle B\rangle$ by Lemma 3.12
(4) $M' \in SN$ by Lemma 3.13
(5) $\mathsf{Cut}(\langle a\rangle M', (x)N') \in SN$ by induction, the degree of the cut-formula is equal in both terms, but $l(M') < l(M)$

In the following we show the cases where a reduction occurs on the top-level.

Commuting Reduction:

(1) $\mathsf{Cut}(\langle a\rangle M, (x)N) \xrightarrow{c'} M\{a := (x)N\}$ and $\langle a{:}B\rangle M \in \langle B\rangle$ by assumption

We know that the commuting reduction is only applicable, if M does not freshly introduce a. This implies that there are only two possibilities for $\langle a{:}B\rangle M$ to be in $\langle B\rangle$: it can be in $\text{AXIOMS}_{\langle B\rangle}$ or in $\text{BINDING}_{\langle B\rangle}((B))$.

In the first case M is an axiom that does not introduce a. Thus $M\{a := (x)N\}$ is equivalent to M, which we know is strongly normalising by assumption.

The proof for the second case is as follows.

(2) $\langle a{:}B\rangle M \in \text{BINDING}_{\langle B\rangle}((B))$ new assumption
(3) $M\{a := (y{:}B)P\} \in SN$ for all $(y{:}B)P \in (B)$ by Definition 3.3
(4) $(x{:}B)N \in (B)$ by assumption
(5) $M\{a := (x{:}B)N\} \in SN$ by (3), (4) and $(y)P \equiv (x)N$

The case where $\mathsf{Cut}(\langle a\rangle M, (x)N)$ reduces to $N\{x := \langle a\rangle M\}$ is analogous. It remains to check for every logical cut-reduction rule that the immediate reducts of $\mathsf{Cut}(\langle a\rangle M, (x)N)$ are strongly normalising. Here we give just three cases to illustrate the proof. The difficult case is the logical reduction $\supset_R/\supset_L$, because both immediate reducts have two nested cuts.

Logical Reduction with Axioms: $\mathsf{Cut}(\langle a\rangle \mathsf{Ax}(y,a), (x)N)$ reduces to $N[x \mapsto y]$. By assumption we know that N is strongly normalising, and therefore $N[x \mapsto y]$ must be strongly normalising.

Logical Reduction $\wedge_R/\wedge_{L_1}$, $B \equiv C \wedge D$:

(1) $\mathsf{Cut}(\langle c\rangle M, (y)N) \xrightarrow{l} \mathsf{Cut}(\langle a\rangle S, (x)U)$, $M \equiv \mathsf{And}_R(\langle a\rangle S, \langle b\rangle T, c)$, $N \equiv \mathsf{And}^1_L((x)U, y)$, M and N freshly introduce c and y, respectively,
$\langle c{:}C\wedge D\rangle M \in \langle C\wedge D\rangle$ and $(y{:}C\wedge D)N \in (C\wedge D)$ by assumption

(2) By Lemma 3.6(i) we have:

$$\langle c{:}C\wedge D\rangle M \in \text{BINDING}_{\langle C\wedge D\rangle}((C\wedge D)) \cup \text{ANDRIGHT}_{\langle C\wedge D\rangle}(\langle C\rangle, \langle D\rangle)$$
$$(y{:}C\wedge D)N \in \text{BINDING}_{(C\wedge D)}(\langle C\wedge D\rangle) \cup \text{ANDLEFT}^1_{(C\wedge D)}((C))$$

Now our argument splits into two cases depending on whether at least one of the $\langle c{:}C\wedge D\rangle M$ and $(y{:}C\wedge D)N$ belong to BINDING. Let us assume $\langle c{:}C\wedge D\rangle M$ is in $\text{BINDING}_{\langle C\wedge D\rangle}((C\wedge D))$.

(3.1) $\langle c{:}C\wedge D\rangle M \in \text{BINDING}_{\langle C\wedge D\rangle}((C\wedge D))$ new assumption
(3.2) $M\{c := (z)P\} \in SN$ for all $(z{:}C\wedge D)P \in (C\wedge D)$ by Definition 3.3
(3.3) $M\{c := (y)N\} \in SN$ by (1), (3.2) and $(z)P \equiv (y)N$
(3.4) The following calculation shows that $\mathsf{Cut}(\langle c\rangle M, (y)N) \in SN$.

$$\begin{aligned} M\{c := (y)N\} &\equiv \mathsf{And}_R(\langle a\rangle S, \langle b\rangle T, c)\{c := (y)N\} \\ &= \mathsf{Cut}(\langle c\rangle \mathsf{And}_R(\langle a\rangle S\{c := (y)N\}, \langle b\rangle T\{c := (y)N\}, c), (y)N) \\ &\equiv \mathsf{Cut}(\langle c\rangle \mathsf{And}_R(\langle a\rangle S, \langle b\rangle T, c), (y)N) \qquad \text{because } M \text{ freshly introduces } c \\ &\equiv \mathsf{Cut}(\langle c\rangle M, (y)N) \end{aligned}$$

If $\mathsf{Cut}(\langle c\rangle M, (y)N)$ is strongly normalising, then its reduct $\mathsf{Cut}(\langle a\rangle S, (x)U)$ must be strongly normalising, too. In case $(y{:}C\wedge D)N$ is in $\text{BINDING}_{(C\wedge D)}(\langle C\wedge D\rangle)$, we reason analogous.

If neither $\langle c{:}C\wedge D\rangle M$ nor $(y{:}C\wedge D)N$ are in BINDING, then we proceed as follows.

(4.1) $\langle c{:}C\wedge D\rangle M \in \text{ANDRIGHT}_{\langle C\wedge D\rangle}(\langle C\rangle, \langle D\rangle)$,
$(y{:}C\wedge D)N \in \text{ANDLEFT}^1_{(C\wedge D)}((C))$ new assumption
(4.2) $\langle a : C\rangle S \in \langle C\rangle$ and $(x : C)U \in (C)$ by Definition 3.3
(4.3) S and U are strongly normalising by Lemma 3.13
(4.4) $\mathsf{Cut}(\langle a{:}C\rangle S, (x{:}C)U) \in SN$ by induction (the degree decreased)

Logical Reduction $\supset_R/\supset_L$, $B \equiv C \supset D$:

(1) $M \equiv \mathsf{Imp}_R((x)\langle a\rangle S, b)$, $N \equiv \mathsf{Imp}_L(\langle c\rangle T, (y)U, z)$,
M and N freshly introduce b and z, respectively,
$\langle c{:}C \supset D\rangle M \in \langle C \supset D\rangle$ and $(y{:}C \supset D)N \in (C \supset D)$ by assumption

The term $\mathsf{Cut}(\langle b\rangle M, (z)N)$ reduces to either

$$\mathsf{Cut}(\langle a\rangle\ \mathsf{Cut}(\langle c\rangle T, (x)S), (y)U) \qquad \text{or} \qquad \mathsf{Cut}(\langle c\rangle T, (x)\ \mathsf{Cut}(\langle a\rangle S, (y)U))\ .$$

We have to show that both reducts are strongly normalising. We shall however only analyse the first case in detail.

(2) By Lemma 3.6(i) we have:

$$\langle b{:}C \supset D\rangle M \in \textsc{binding}_{\langle C\supset D\rangle}((C \supset D)) \ \cup\ \textsc{impright}_{\langle C\supset D\rangle}((C), \langle D\rangle)$$
$$(z{:}C \supset D)N \in \textsc{binding}_{(C\supset D)}(\langle C \supset D\rangle) \ \cup\ \textsc{impleft}_{(C\supset D)}(\langle C\rangle, (D))$$

Again the proof splits into two cases depending on whether the co-named term $\langle b : C \supset D\rangle M$ or the named term $(z : C \supset D)N$ belong to BINDING. Let us assume $\langle b{:}C \supset D\rangle M$ is an element in $\textsc{binding}_{\langle C\supset D\rangle}((C \supset D))$.

(3.1) $\langle b{:}C \supset D\rangle M \in \textsc{binding}_{\langle C\supset D\rangle}((C \supset D))$ new assumption
(3.2) $M\{b := (v)P\} \in SN$ for all $(v{:}C \supset D)P \in (C \supset D)$ by Definition 3.3
(3.3) $M\{b := (z)N\} \in SN$ by (1), (3.2) and $(v)P \equiv (z)N$
(3.4) The following calculation shows that $\mathsf{Cut}(\langle c\rangle M, (y)N) \in SN$.

$$\begin{aligned} M\{b := (z)N\} &\equiv \mathsf{Imp}_R((x)\langle a\rangle S, b)\{b := (z)N\} \\ &= \mathsf{Cut}(\langle b\rangle \mathsf{Imp}_R((x)\langle a\rangle S\{b := (z)N\}, b), (z)N) \\ &\equiv \mathsf{Cut}(\langle b\rangle \mathsf{Imp}_R((x)\langle a\rangle S, b), (z)N) \qquad \text{because } M \text{ freshly introduces } b \\ &\equiv \mathsf{Cut}(\langle b\rangle M, (z)N) \end{aligned}$$

Therefore we know that the term $\mathsf{Cut}(\langle b\rangle M, (z)N)$ is strongly normalising, and hence its reduct $\mathsf{Cut}(\langle a\rangle\ \mathsf{Cut}(\langle c\rangle T, (x)S), (y)U)$ must be strongly normalising, too. In fact both reducts must be strongly normalising. The case where $(z{:}C \supset D)N$ belongs to $\textsc{binding}_{(C\supset D)}(\langle C \supset D\rangle)$ is similar.

We now have to show that the reduct is strongly normalising in the case where $\langle b : C \supset D\rangle M$ and $(z : C \supset D)N$ belong to $\textsc{impright}_{\langle C\supset D\rangle}((C), \langle D\rangle)$ and to $\textsc{impleft}_{(C\supset D)}(\langle C\rangle, (D))$, respectively.

We first show that the inner cut of the reduct is strongly normalising.

(4.1) $\langle b : C \supset D\rangle M \in \textsc{impright}_{\langle C\supset D\rangle}((C), \langle D\rangle)$,
$(z : C \supset D)N \in \textsc{impleft}_{(C\supset D)}(\langle C\rangle, (D))$ new assumption
(4.2) $(x : C)S\{a := (v{:}D)P\} \in (C)$ for all $(v{:}D)P \in (D)$,
$\langle c{:}C\rangle T \in \langle C\rangle$ by Definition 3.3
(4.3) $S\{a := (v{:}D)P\} \in SN$ and $T \in SN$ by Lemma 3.13
(4.4) $\mathsf{Cut}(\langle c\rangle T, (x)S\{a := (v{:}D)P\}) \in SN$ by induction (the degree decreased)

We required that a is not free in $\langle c\rangle T$ (see Remark 2.3), and therefore we may move the substitution on the top-level. Thus we have that

$$\mathsf{Cut}(\langle c\rangle T, (x)S\{a := (v{:}D)P\}) \equiv \mathsf{Cut}(\langle c\rangle T, (x)S)\{a := (v{:}D)P\}.$$

(4.5) $\mathsf{Cut}(\langle c\rangle T, (x)S)\{a := (v{:}D)P\} \in SN$ for all $(v{:}D)P \in (D)$ by (4.4)
(4.6) $\langle a{:}D\rangle\mathsf{Cut}(\langle c\rangle T, (x)S) \in \langle D\rangle$ by Definition 3.3

Now we show that the outer cut is strongly normalising.

(4.7) $(y{:}D)U \in (D)$ by (4.1) and Definition 3.3
(4.8) $\mathsf{Cut}(\langle c\rangle T, (x)S) \in SN$ and $U \in SN$ by Lemma 3.13
(4.9) $\mathsf{Cut}(\langle a\rangle\mathsf{Cut}(\langle c\rangle T, (x)S), (y)U) \in SN$ by induction (the degree decreased)

We reason analogous in the case where

$$\mathsf{Cut}(\langle b\rangle M, (z)N) \xrightarrow{l} \mathsf{Cut}(\langle c\rangle T, (x)\ \mathsf{Cut}(\langle a\rangle S, (y)U))\ .$$

We have shown that all immediate reducts of $\mathsf{Cut}(\langle a\rangle M, (x)N)$ are strongly normalising. Consequently, $\mathsf{Cut}(\langle a\rangle M, (x)N)$ must be strongly normalising. Thus we are done. □

It is left to show that all well-typed terms are strongly normalising. In order to do so, we shall consider a special class of substitutions, which are called *safe*. Two substitutions, say $\{\sigma\}$ and $\{\tau\}$, are safe, if and only if the domain of $\{\sigma\}$ is not free in the co-domain of $\{\tau\}$ and the domain of $\{\tau\}$ is not free in the co-domain of $\{\sigma\}$. For example $\{x := \langle a\rangle P\}$ and $\{b := (y)Q\}$ are safe provided that x is not free in $(y)Q$ and b is not free in $\langle a\rangle P$. As explained earlier, the auxiliary substitution operation, $\{\_\}$, is defined with the property in mind that safe substitutions can commute. A special case of the substitution lemma for $\{\_\}$ (Lemma 3.8) ensures that for all M and any two safe substitutions $\{\sigma\}$ and $\{\tau\}$ we have

$$M\{\sigma\}\{\tau\} \equiv M\{\tau\}\{\sigma\}\ .$$

We shall now extend the notion of safety from substitutions to simultaneous substitutions; that is to sets of substitutions.

Definition 3.15. (Safe Simultaneous Substitution, *sss*)

$\emptyset$ is an *sss*.
$\hat{\sigma} \cup \{\sigma\}$ is an *sss*, if and only if $\hat{\sigma}$ is an *sss*, $dom(\{\sigma\}) \notin dom(\hat{\sigma})$, $dom(\{\sigma\})$ not free in $codom(\hat{\sigma})$, and $dom(\hat{\sigma})$ not free in $codom(\{\sigma\})$.

□

In the presence of our Barendregt-style naming convention and alpha-conversion, any set of substitutions can be transformed into a safe simultaneous substitution. We shall, however, omit a formal proof and rather give the reader the following example.

Example 3.16. Suppose we have a term, say M, and a safe simultaneous substitution, say

$$\hat{\sigma} \equiv \Big\{\{x := \langle c\rangle\mathsf{Ax}(x, b)\}, \{a := (z)\mathsf{Ax}(z, c)\}\Big\},$$

and let us assume we have the substitution $\{\sigma\} \equiv \{b := (y)\mathsf{Ax}(x, a)\}$. Clearly, $\hat{\sigma} \cup \{\sigma\}$ is not safe, and therefore $M(\hat{\sigma} \cup \{\sigma\})$ is an ill-defined expression. However, x, a and b are considered as binders and thus can be rewritten. In effect, we can form the following safe version of $\hat{\sigma} \cup \{\sigma\}$

$$\hat{\sigma}_{\mathit{safe}} = \Big\{\{x' := \langle c\rangle\mathsf{Ax}(x, b)\}, \{a' := (z)\mathsf{Ax}(z, c)\}, \{b' := (y)\mathsf{Ax}(x, a)\}\Big\},$$

assuming that x', a' and b' are fresh. Subsequently, we need to rewrite the corresponding names and co-names in M. Now the expression $M[x \mapsto x'][a \mapsto a'][b \mapsto b']\,\hat{\sigma}_{\mathit{safe}}$ is well-defined. □

In the next lemma, we shall show that a specific substitution built up by axioms is an sss.

Lemma 3.17. Let $\hat{\sigma}$ be of the form

$$\Big\{\bigcup_{i=0,\ldots,n}\{x_i := \langle c\rangle\mathsf{Ax}(x_i, c)\}\Big\} \cup \Big\{\bigcup_{j=0,\ldots,m}\{a_j := (y)\mathsf{Ax}(y, a_j)\}\Big\}$$

where the x_i's and a_i's are distinct names and co-names, respectively. Substitution $\hat{\sigma}$ is an sss.

Proof: By induction on the length of $\hat{\sigma}$. □

Now we can show that every well-typed term together with a closing substitution is strongly normalising. This is again a rather lengthy proof.

Lemma 3.18.

- For every well-typed term M—not necessarily strongly normalising—with a typing judgement $\Gamma \triangleright M \triangleright \Delta$, and
- for every sss, $\hat{\sigma}$, such that $dom(\Gamma) \cup dom(\Delta) \subseteq dom(\hat{\sigma})$, i.e., $\hat{\sigma}$ is a closing substitution,[1] and
  - for every $(x{:}B)P \in codom(\hat{\sigma})$ we require that $(x{:}B)P \in (B)$ and
  - for every $\langle a{:}C\rangle Q \in codom(\hat{\sigma})$ we require that $\langle a{:}C\rangle Q \in \langle C\rangle$;

  we have $M\hat{\sigma} \in SN$.

Proof: The proof proceeds by induction over the structure of M. We shall give four representative cases, in which we write $\hat{\sigma}, \{\sigma\}$ for the set $\hat{\sigma} \cup \{\sigma\}$ and assume $\{\sigma\} \notin \hat{\sigma}$.

Case $\mathsf{Ax}(x, a)$**:** We have to prove that $\mathsf{Ax}(x, a)\ \hat{\sigma}, \{x := \langle b\rangle P\}, \{a := (y)Q\}$ is strongly normalising for arbitrary (co-)named terms $\langle b{:}B\rangle P \in \langle B\rangle$ and $(y{:}B)Q \in (B)$.

(1)	$\mathsf{Ax}(x, a)\ \hat{\sigma}, \{x := \langle b\rangle P\}, \{a := (y)Q\} = \mathsf{Cut}(\langle b\rangle P, (y)Q)$	by Definition of $\{\_\}$
(2)	$\langle b{:}B\rangle P \in \langle B\rangle$ and $(y{:}B)Q \in (B)$	by assumption
(3)	$P \in SN$ and $Q \in SN$	by Lemma 3.13
(4)	$\mathsf{Cut}(\langle b\rangle P, (y)Q) \in SN$	by Lemma 3.14
(5)	$\mathsf{Ax}(x, a)\ \hat{\sigma}, \{x := \langle b\rangle P\}, \{a := (y)Q\} \in SN$	by (1) and (4)

Case $\mathsf{And}_R(\langle a\rangle M, \langle b\rangle N, c)$**:** We have to prove that $\mathsf{And}_R(\langle a\rangle M, \langle b\rangle N, c)\ \hat{\sigma}, \{c := (z)R\}$ is strongly normalising for an arbitrary named term $(z{:}B \wedge C)R \in (B \wedge C)$.

[1] All free names and co-names of M are amongst the domain of $\hat{\sigma}$.

(1) $\mathsf{And}_R(\langle a\rangle M, \langle b\rangle N, c)\,\hat{\sigma}, \{c{:=}(z)R\} =$
$\mathsf{Cut}(\langle c\rangle \mathsf{And}_R(\langle a\rangle\, M\hat{\sigma}, \{c := (z)R\}, \langle b\rangle\, N\hat{\sigma}, \{c := (z)R\}, c), (z)R)$ Definition of $\{\_\}$

(2) $M\ \hat{\sigma}, \{c := (z)R\}, \{a := (y)P\} \in SN$ for arbitrary $(y{:}B)P \in (B)$,
$N\ \hat{\sigma}, \{c := (z)R\}, \{b := (x)Q\} \in SN$ for arbitrary $(x{:}C)Q \in (C)$ by induction

(3) $(M\hat{\sigma}, \{c := (z)R\})\{a := (y)P\} \in SN$,
$(N\hat{\sigma}, \{c := (z)R\})\{b := (x)Q\} \in SN$ by (2) and sss

(4) $\langle a{:}B\rangle(M\hat{\sigma}, \{c := (z)R\}) \in \langle B\rangle$,
$\langle b{:}C\rangle(N\hat{\sigma}, \{c := (z)R\}) \in \langle C\rangle$ by Definition 3.3

(5) $\mathsf{And}_R(\langle a\rangle M\hat{\sigma}\,\{c := (z)R\}, \langle b\rangle N\hat{\sigma}\,\{c := (z)R\}, c)$ freshly introduces c by (1)

(6) $\langle c{:}B\wedge C\rangle\mathsf{And}_R(\langle a\rangle M\hat{\sigma}, \{c := (z)R\}, \langle b\rangle N\hat{\sigma}, \{c := (z)R\}, c) \in \langle B\wedge C\rangle$
by (4), (5) and Definition 3.3

(7) $\mathsf{And}_R(\langle a\rangle M\hat{\sigma}, \{c := (z)R\}, \langle b\rangle N\hat{\sigma}, \{c := (z)R\}, c) \in SN$, $R \in SN$ by Lemma 3.13

(8) $\mathsf{Cut}(\langle c\rangle \mathsf{And}_R(\langle a\rangle M\hat{\sigma}, \{c := (z)R\}, \langle b\rangle N\hat{\sigma}, \{c := (z)R\}, c), (z)R) \in SN$ by Lemma 3.14

(9) $\mathsf{And}_R(\langle a\rangle M, \langle b\rangle N, c)\ \hat{\sigma}, \{c := (z)R\} \in SN$ by (1) and (8)

Case $\mathsf{Imp}_R((x)\langle a\rangle M, b)$**:** We prove that $\mathsf{Imp}_R((x)\langle a\rangle M, b)\ \hat{\sigma}, \{b := (z)R\}$ is strongly normalising for an arbitrary named term $(z{:}B\supset C)R \in (B\supset C)$.

(1) $\mathsf{Imp}_R((x)\langle a\rangle M, b)\ \hat{\sigma}, \{b := (z)R\} =$
$\mathsf{Cut}(\langle c\rangle \mathsf{Imp}_R((x)\langle a\rangle M\hat{\sigma}, \{b := (z)R\}, b), (z)R)$ by Definition of $\{\_\}$

(2) $M\ \hat{\sigma}, \{b := (z)R\}, \{a := (y)P\}, \{x := \langle c\rangle Q\} \in SN$
for arbitrary $(y{:}B)P \in (B)$ and $\langle c{:}C\rangle Q \in \langle C\rangle$ by induction

(3) $(M\hat{\sigma}, \{b := (z)R\}, \{x := \langle c\rangle Q\})\{a := (y)P\} \in SN$,
$(M\hat{\sigma}, \{b := (z)R\}, \{a := (y)P\})\{x := \langle c\rangle Q\} \in SN$ by (2) and sss

(4) $\langle a{:}B\rangle\ M\hat{\sigma}, \{b := (z)R\}, \{x := \langle c\rangle Q\} \in \langle B\rangle$
$(x{:}C)\ M\hat{\sigma}, \{b := (z)R\}, \{a := (y)P\} \in (C)$ by Definition 3.3

(5) $\mathsf{Imp}_R((x)\langle a\rangle M\hat{\sigma}, \{b := (z)R\}, b)$ freshly introduces b by (1)

(6) $\langle b{:}B\supset C\rangle\mathsf{Imp}_R((x)\langle a\rangle M\hat{\sigma}, \{b := (z)R\}, b) \in \langle B\supset C\rangle$ by (4) and (5) and Definition 3.3

(7) $\mathsf{Imp}_R((x)\langle a\rangle M\hat{\sigma}, \{b := (z)R\}, b) \in SN$ and $R \in SN$ by Lemma 3.13

(8) $\mathsf{Cut}(\langle b\rangle \mathsf{Imp}_R((x)\langle a\rangle M\hat{\sigma}, \{b := (z)R\}, b), (z)R) \in SN$ by Lemma 3.14

(9) $\mathsf{Imp}_R((x)\langle a\rangle M, b)\ \hat{\sigma}, \{b := (z)R\} \in SN$ by (1) and (8)

Case $\mathsf{Cut}(\langle a\rangle M, (x)N)$**:** Since we introduced in the definition of $\{\_\}$ two special clauses for cuts with an axiom as immediate subterm, we have to distinguish two cases.

Subcase I: M is an axiom that freshly introduces the label of the cut-formula (the case N being such an axiom is similar). We have to show that for arbitrary $\langle b{:}B\rangle R \in \langle B\rangle$ the term $\mathsf{Cut}(\langle a\rangle \mathsf{Ax}(x, a), (y)N)\ \hat{\sigma}, \{x := \langle b\rangle R\}$ is strongly normalising.

(1) $\mathsf{Cut}(\langle a\rangle \mathsf{Ax}(x,a), (y)N)\ \hat{\sigma}, \{x := \langle b\rangle R\} =$
$\mathsf{Cut}(\langle b\rangle R, (y)\ N\hat{\sigma}, \{x := \langle b\rangle R\})$ by Definition of $\{\_\}$

(2) $N\ \hat{\sigma}, \{x := \langle b\rangle R\}, \{y := \langle c\rangle P\} \in SN$ for arbitrary $\langle c{:}B\rangle P \in \langle B\rangle$ by induction

(3) $(N\ \hat{\sigma}, \{x := \langle b\rangle R\})\{y := \langle c\rangle P\} \in SN$ by (2) and sss

(4) $(y{:}\ B)\ N\hat{\sigma}, \{x := \langle b\rangle R\} \in (B)$ by Definition 3.3

(5) $N\ \hat{\sigma}, \{x := \langle b\rangle R\} \in SN$ and $R \in SN$ by Lemma 3.13

(6) $\mathsf{Cut}(\langle b\rangle R, (y)\ N\hat{\sigma}, \{x := \langle b\rangle R\}) \in SN$ by Lemma 3.14

(7) $\mathsf{Cut}(\langle a\rangle \mathsf{Ax}(x,a), (y)N)\ \hat{\sigma}, \{x := \langle b\rangle R\} \in SN$ by (1) and (6)

Subcase II: M and N are not axioms that freshly introduce the label of the cut-formula. We have to prove that $\mathsf{Cut}(\langle a\rangle M, (x)N)\ \hat{\sigma}$ is strongly normalising.

(1) $\mathsf{Cut}(\langle a\rangle M, (x)N)\ \hat{\sigma} = \mathsf{Cut}(\langle a\rangle M\hat{\sigma}, (x)N\hat{\sigma})$ by Definition of $\{\_\}$

(2) $M\ \hat{\sigma}, \{a := (y)S\} \in SN$ for arbitrary $(y{:}B)S \in (B)$,
$N\ \hat{\sigma}, \{x := \langle b\rangle T\} \in SN$ for arbitrary $\langle b{:}B\rangle T \in \langle B\rangle$ by induction

(3) $(M\hat{\sigma})\{a := (y)S\} \in SN$ and $(N\hat{\sigma})\{x := \langle b\rangle T\} \in SN$ by (2) and sss

(4) $\langle a{:}B\rangle\ M\hat{\sigma} \in \langle B\rangle$ and $(x{:}B)\ N\hat{\sigma} \in (B)$ by Definition 3.3

(5) $M\hat{\sigma} \in SN$ and $N\hat{\sigma} \in SN$ by Lemma 3.13

(6) $\mathsf{Cut}(\langle a\rangle\ M\hat{\sigma}, (x)\ N\hat{\sigma}) \in SN$ by Lemma 3.14

(7) $\mathsf{Cut}(\langle a\rangle M, (x)N)\ \hat{\sigma} \in SN$ by (1) and (6)

□

We are now able to prove that $(\mathcal{T}, \xrightarrow{aux})$ is strongly normalising.

Theorem 3.19. For all well-typed terms $\xrightarrow{aux}$ is strongly normalising.

Proof: By Lemma 3.18 we know that for an arbitrary well-typed term, say M, with the typing judgement $\Gamma \triangleright M \triangleright \Delta$ and an arbitrary safe simultaneous substitution, say $\hat{\sigma}$, where all free names and co-names are amongst the domain of σ, the term $M\hat{\sigma}$ is strongly normalising. Taking $\hat{\sigma}$ to be the safe simultaneous substitution from Lemma 3.17 we can infer, using Lemma 3.10, that $M\hat{\sigma} \xrightarrow{aux}{}^{*}\ M$, and we therefore have that M is strongly normalising. Thus we are done. □

From this result we can deduce strong normalisation for $(\mathcal{T}, \xrightarrow{cut})$, which is relatively straightforward since every $\xrightarrow{cut}$-reduction maps onto a series of $\xrightarrow{aux}$-reductions. First we prove that $M\{\sigma\}$ reduces to or is equivalent to $M[\sigma]$.

Lemma 3.20. For all $M, N \in \mathcal{T}$ we have

(i) $M\{a := (x)N\} \xrightarrow{aux}{}^{*}\ M[a := (x)N]$

(ii) $M\{x := \langle a\rangle N\} \xrightarrow{aux}{}^{*}\ M[x := \langle a\rangle N]$

Proof: By induction on the structure of M. The non-trivial cases are where the definitions of $\{\_\}$ and $[\_]$ differ, one of which is given below.

Case $M \equiv \mathsf{Cut}(\langle b\rangle P, (y)\mathsf{Ax}(y,a))$:

$$
\begin{array}{rcl}
M\{a := (x)N\} & \equiv & \mathsf{Cut}(\langle b\rangle P, (y)\mathsf{Ax}(y,a))\{a := (x)N\} \\
 & = & \mathsf{Cut}(\langle b\rangle\ P\{a := (x)N\}, (x)N) \\
 & \stackrel{aux}{\underset{\text{(by IH)}}{\longrightarrow}}{}^{*} & \mathsf{Cut}(\langle b\rangle\ P[a := (x)N], (x)N) \\
M[a := (x)N] & \equiv & \mathsf{Cut}(\langle b\rangle P, (y)\mathsf{Ax}(y,a))[a := (x)N] \\
 & = & \mathsf{Cut}(\langle b\rangle\ P[a := (x)N], (y)\ \mathsf{Ax}(y,a)[a := (x)N]) \\
 & = & \mathsf{Cut}(\langle b\rangle\ P[a := (x)N], (y)\ N[x \mapsto y]) \\
 & \stackrel{(*)}{\equiv} & \mathsf{Cut}(\langle b\rangle\ P[a := (x)N], (x)N)
\end{array}
$$

$(*)$ because by the Barendregt-style naming convention y cannot be free in N. □

The next lemma shows that every $\xrightarrow{cut}$-reduction maps onto a series of $\xrightarrow{aux}$-reductions.

Lemma 3.21. For all $M, N \in \mathcal{T}$, if $M \xrightarrow{cut} N$, then $M \xrightarrow{aux}{}^{+} N$.

Proof: By induction on the structure of $\xrightarrow{cut}$. We analyse all possible cases of $\xrightarrow{cut}$-reductions.

Inner Reduction: Given that $M \xrightarrow{cut} N$, there is a proper subterm in M, say S, which reduces to S'. This term S' is a subterm of N. We know by induction that $S \xrightarrow{aux}{}^{+} S'$ and by context closure that $M \xrightarrow{aux}{}^{+} N$.

Logical Reduction: This case is obvious, because both $\xrightarrow{cut}$ and $\xrightarrow{aux}$ perform the same logical reductions.

Commuting Reduction: Suppose $M \xrightarrow{c} N$ with $M \equiv \mathsf{Cut}(\langle a\rangle S, (x)T)$ and $N \equiv S[a := (x)T]$, then we know that $M \xrightarrow{c'} S\{a := (x)T\}$. From Lemma 3.20 we have that $S\{a := (x)T\} \xrightarrow{aux}{}^{*} N$, and therefore $M \xrightarrow{aux}{}^{+} N$. The symmetric case is analogous.

□

Now it is rather easy to show that $(\mathcal{T}, \xrightarrow{cut})$ is strongly normalising

Theorem 3.22. (Main Theorem)
For all well-typed terms $\xrightarrow{cut}$ is strongly normalising.

Proof: Since $\xrightarrow{aux}$ is strongly normalising for all well-typed terms, and whenever $M \xrightarrow{cut} N$, we have that $M \xrightarrow{aux}{}^{+} N$. Consequently, the reduction $\xrightarrow{cut}$ must be strongly normalising. □

4. First-Order Classical Logic

In this section we extend the cut-elimination procedure $(\mathcal{T}, \xrightarrow{cut})$ to the first-order fragment of classical logic. To do so, we extend the notion of a formula by allowing atomic formulae to have arguments ranging over *expressions* and introduce the quantifiers $\forall$ and $\exists$. We use 'expression' instead of 'term'—the standard terminology—in order to avoid confusion with our terminology introduced in previous sections. Moreover, we use a sans serif font for expressions to clearly distinguish them from terms. The grammar for expressions is

$$\mathsf{t} ::= \mathsf{x} \mid \mathsf{f}\,\mathsf{t}\ldots\mathsf{t}$$

where x is taken from a set of *variables* and f from a set of functional symbols. The formulae in the first-order case are given by the grammar

$$B ::= A\mathsf{t}_1\ldots\mathsf{t}_\mathsf{n} \mid \neg B \mid B\wedge B \mid B\vee B \mid B\supset B \mid \forall\mathsf{x}.B \mid \exists\mathsf{x}.B$$

where A ranges over predicate symbols, each of which take a fixed number of expressions as arguments, and where x ranges over variables.

Instances of the cut-rule with a quantified formula as cut-formula are eliminated such that in a subproof a variable, also called *eigenvariable*, is replaced by an expression. At the level of terms this means some term constructors introduce an expression, which may appear in a substitution for an eigenvariable; other term constructors bind a variable to signify that it is a placeholder for which an expression may be substituted. The set of raw terms, $\mathcal{R}^{\forall\exists}$, is obtained by extending the grammar of $\mathcal{R}$ given in Section 2 with the clauses

$$\begin{array}{lcll} M, N & ::= & \mathsf{Forall}_R(\langle a{:}B\rangle[\mathsf{y}]M, b) & \text{Forall-R}\\ & \mid & \mathsf{Forall}_L((x{:}B)M, \mathsf{t}, y) & \text{Forall-L}\\ & \mid & \mathsf{Exists}_R(\langle a{:}B\rangle M, \mathsf{t}, b) & \text{Exists-R}\\ & \mid & \mathsf{Exists}_L((x{:}B)[\mathsf{y}]M, y) & \text{Exists-L}\end{array}$$

in which y is a variable and t is an expression. In these terms, square brackets indicate that a variable becomes bound, and analogous to names and co-names we observe a Barendregt-style naming convention for bound and free variables.

We now give the typing rules for the new terms and the corresponding cut-reductions. In both we shall use the standard notion of (capture avoiding) variable substitution, written as $[\mathsf{x} := \mathsf{t}]$; this notion is defined over expressions, formulae and terms. The are four typing rules to govern the new term constructors.

$$\frac{x : B[\mathsf{x} := \mathsf{t}], \Gamma \vartriangleright M \vartriangleright \Delta}{y : \forall\mathsf{x}.B, \Gamma \vartriangleright \mathsf{Forall}_L((x)M, \mathsf{t}, y) \vartriangleright \Delta}\ \forall_L \qquad \frac{\Gamma \vartriangleright M \vartriangleright \Delta, a : B[\mathsf{x} := \mathsf{y}]}{\Gamma \vartriangleright \mathsf{Forall}_R(\langle a\rangle[\mathsf{y}]M, b) \vartriangleright \Delta, b : \forall\mathsf{x}.B}\ \forall_R$$

$$\frac{x : B[\mathsf{x} := \mathsf{y}], \Gamma \vartriangleright M \vartriangleright \Delta}{y : \exists\mathsf{x}.B, \Gamma \vartriangleright \mathsf{Exists}_L((x)[\mathsf{y}]M, y) \vartriangleright \Delta}\ \exists_L \qquad \frac{\Gamma \vartriangleright M \vartriangleright \Delta, a : B[\mathsf{x} := \mathsf{t}]}{\Gamma \vartriangleright \mathsf{Exists}_R(\langle a\rangle M, \mathsf{t}, b) \vartriangleright \Delta, b : \exists\mathsf{x}.B}\ \exists_R$$

The $\forall_R$ and $\exists_L$ rules are subject to the usual proviso that y does not appear free in Γ and Δ.
The cut-reductions are given next.

Definition 4.1. (Cut-Reductions for Quantifiers)

$$\mathsf{Cut}(\langle b\rangle\mathsf{Forall}_R(\langle a\rangle[\mathsf{y}]M, b), (y)\mathsf{Forall}_L((x)N, \mathsf{t}, y)) \xrightarrow{l} \mathsf{Cut}(\langle a\rangle M[\mathsf{y} := \mathsf{t}], (x)N)$$
$$\text{if } \mathsf{Forall}_R(\langle a\rangle[\mathsf{y}]M, b) \text{ and } \mathsf{Forall}_L((x)N, \mathsf{t}, y) \text{ freshly introduce } a \text{ and } x$$

$$\mathsf{Cut}(\langle b\rangle\mathsf{Exists}_R(\langle a\rangle M, \mathsf{t}, b), (y)\mathsf{Exists}_L((x)[\mathsf{y}]N, y)) \xrightarrow{l} \mathsf{Cut}(\langle a\rangle M, (x)N[\mathsf{y} := \mathsf{t}])$$
$$\text{if } \mathsf{Exists}_R(\langle a\rangle M, \mathsf{t}, b) \text{ and } \mathsf{Exists}_L((x)[\mathsf{y}]N, y) \text{ freshly introduce } a \text{ and } x$$

□

We identify, as usual, terms which differ only in their names of bound variables, because otherwise the cut-reductions given above are not complete, in that some instances of the cut-rule cannot be eliminated. An example of an LK-proof where a renaming of a variable is required during cut-elimination is given in [19, 24]. In our calculus we assume that these renamings are done implicitly.

The definition of our proof substitution (see Figure 3) and the reduction $\xrightarrow{c}$ can be extended to $\mathfrak{R}^{\forall\exists}$ in the obvious way. Therefore the details are omitted. Our strong normalisation proof given for the propositional fragment of classical logic can be extended to include the rules dealing with the quantifiers; for space reasons, however, we only give the set operators for the universal quantifier and omit the other details.

$$\text{FORALLLEFT}_{(\forall \mathsf{x}.B)}(X) \quad \stackrel{\text{def}}{=} \quad \left\{ \begin{array}{l} (y{:}\forall \mathsf{x}.B)\mathsf{Forall}_L((x{:}B[\mathsf{x} := \mathsf{t}])M, \mathsf{t}, y) \mid \\ \qquad \mathsf{Forall}_L((x{:}B[\mathsf{x} := \mathsf{t}])M, \mathsf{t}, y) \text{ freshly introduces } y, \\ \qquad (x{:}B[\mathsf{x} := \mathsf{t}])M \in X \end{array} \right\}$$

$$\text{FORALLRIGHT}_{\langle \forall \mathsf{x}.B \rangle}(X) \quad \stackrel{\text{def}}{=} \quad \left\{ \begin{array}{l} (b{:}\forall \mathsf{x}.B)\mathsf{Forall}_R(\langle a{:}B[\mathsf{x} := \mathsf{y}]\rangle[\mathsf{y}]M, b) \mid \\ \qquad \mathsf{Forall}_R(\langle a{:}B[\mathsf{x} := \mathsf{y}]\rangle[\mathsf{y}]M, b) \text{ freshly introduces } b, \\ \qquad \text{for all } \mathsf{t} \;\; (x{:}B[\mathsf{x} := \mathsf{t}])M \in X \end{array} \right\}$$

5. Conclusion

In this paper we have shown that only a slight reformulation of the standard cut-reductions is sufficient to obtain a strongly normalising cut-elimination procedure for classical logic. Prior to our work some strongly normalising cut-elimination procedures have been developed, but all of them impose some quite strong restrictions. In consequence, all of them violate one or more criteria we put forward in the Introduction. In particular, they restrict the normal forms reachable from a proof containing cuts, which we feel is unfortunate as the normal forms play an important rôle in investigating the computational content of classical logic. Therefore we have improved the cut-elimination procedure of [9], mainly by removing the restrictions imposed by the colour annotation on formulae.

It is important to notice that our cut-elimination procedure is non-deterministic. Consider Lafont's now famous example [15, Page 151].

$$\dfrac{\dfrac{\pi_1 \left\{ \dfrac{\vdots}{\dfrac{\vdash B}{\vdash B, C}\ \text{Weak}_R} \right. \quad \left. \dfrac{\vdots}{\dfrac{\vdash B}{C \vdash B}\ \text{Weak}_L} \right\} \pi_2}{\vdash B, B}\ \text{Cut}}{\vdash B}\ \text{Contr}_R$$

Rewritten in our system, this proof would be represented by the term $\mathsf{Cut}(\langle a\rangle M, (x)N)$ where M and N correspond to π_1 and π_2, respectively, and where a is not free in M and x is not free in N. According to

our cut-elimination procedure, we may reduce this term to either M or N, non-deterministically.

Another point to notice is that our substitution operation is a global proof transformation. (A transformation is said to be *local*, if only neighbouring inference rules are rewritten possibly duplicating a subderivation; otherwise the transformation is said to be global.) Using results concerning explicit substitution calculi we can replace this global operation with completely local proof transformations and obtain again a strongly normalising cut-elimination procedure. This result allows us to prove strong normalisation for a slight variant of Gentzen's original cut-elimination procedure for LK. The details are given in [25].

While both cut-elimination procedures, $(\mathcal{T}, \xrightarrow{cut})$ and $(\mathcal{T}, \xrightarrow{aux})$, are complete, in the sense that they are effective procedures for eliminating all instances of a cut-rule in a sequent proof, $(\mathcal{T}, \xrightarrow{cut})$ can be used to show strong normalisation for normalisation in natural deduction using the standard translations between natural deduction proofs and sequent proofs as presented, for example, in [27]. This result, too, is given in [25].

Note also that our term annotation encodes precisely the structure of sequent proofs. In consequence, we were able to adopt proof techniques from term rewriting for proving the strong normalisation property of cut-elimination. Pfenning used a similar term annotation for proving in LF the weak normalisation property [20]. A different term annotation, called the symmetric lambda calculus, was introduced by Barbanera and Berardi [2]. Although in the $(\wedge, \vee)$-fragment of classical logic there are simple translations between symmetric lambda terms and our terms, we found that our reduction rules are more general, in that strong normalisation of the symmetric lambda calculus can be inferred from our strong normalisation result, but not vice versa (using a simple translation [25]). Moreover, for implication and multiplicative connectives the symmetric lambda calculus seems very inconvenient, because it does not allow multiple binders. Yet another term annotation, motivated by a study of a specific computational interpretation of classical logic, was introduced by Curien and Herbelin [8]. Independent from our work [26], and others [20], they also proposed to use multiple binders for dealing with the implicational-right rule. Their main finding is that if one consistently gives priority to one of the clauses of $\xrightarrow{c}$ (see Figure 4), then one obtains either a call-by-name or call-by-value lambda calculus. Of course imposing a priority means that the resulting reduction system is deterministic, like the cut-elimination procedure presented for LK^{tq} [9]. While the observation of Curien and Herbelin is interesting, our suspicion is that it does not extend to our non-deterministic setting.

Most of the work concerning the computational interpretation of classical logic focuses on functional programs enriched with control operators, such as operators for continuation passing [4, 5, 8, 16]. We should like to promote the view that the computational content of classical logic can also be seen as non-deterministic computation. A similar view is taken in [2]. However the consequences of this view for programming remain to be investigated. We leave this and a semantical study of our cut-elimination procedure to future work.

Acknowledgements

We should like to thank Roy Dyckhoff and Martin Hyland for their help and encouragement. The work has greatly benefited from discussions with Harold Schellinx and Jean-Baptiste Joinet concerning LK^{tq}. The work was completed whilst Urban was at the University of Cambridge Computer Laboratory and supported by a scholarship from the DAAD. Bierman was supported by EPSRC Grant GR-M04716 and Gonville & Caius College, Cambridge.

References

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

[2] F. Barbanera and S. Berardi. A Symmetric Lambda Calculus for "Classical" Program Extraction. In *Theoretical Aspects of Computer Software*, volume 789 of *LNCS*, pages 495–515. Springer Verlag, 1994.

[3] H. Barendregt and S. Ghilezan. Theoretical Pearls: Lambda Terms for Natural Deduction, Sequent Calculus and Cut Elimination. *Journal of Functional Programming*, 10(1):121–134, 2000.

[4] U. Berger and H. Schwichtenberg. Program Extraction from Classical Proofs. In *Logic and Computational Complexity*, volume 960 of *LNCS*, pages 77–97, 1995.

[5] G. M. Bierman. A Computational Interpretation of the $\lambda\mu$-calculus. In *Mathematical Foundations of Computer Science*, volume 1450 of *LNCS*, pages 336–345. Springer-Verlag, 1998.

[6] E. T. Bittar. Strong Normalisation Proofs for Cut-Elimination in Gentzen's Sequent Calculi. In *Logic, Algebra and Computer Science*, volume 46 of *Banach-Center Publications*, pages 179–225, 1999.

[7] E. A. Cichon, M. Rusinowitch, and S. Selhab. Cut-Elimination and Rewriting: Termination Proofs. Technical Report, 1996.

[8] P.-L. Curien and H. Herbelin. The Duality of Computation. In *Conference on Functional Programming*, pages 233–243. ACM Press, 2000.

[9] V. Danos, J.-B. Joinet, and H. Schellinx. A New Deconstructive Logic: Linear Logic. *Journal of Symbolic Logic*, 62(3):755–807, 1997.

[10] A. G. Dragalin. *Mathematical Intuitionism: Introduction to Proof Theory*, volume 67 of *Translations of Mathematical Monographs*. American Mathematical Society, 1988.

[11] R. Dyckhoff and L. Pinto. Cut-Elimination and a Permutation-Free Sequent Calculus for Intuitionistic Logic. *Studia Logica*, 60(1):107–118, 1998.

[12] J. Gallier. Constructive Logics. Part I: A Tutorial on Proof Systems and Typed λ-calculi. *Theoretical Computer Science*, 110(2):249–239, 1993.

[13] G. Gentzen. Untersuchungen über das logische Schließen I and II. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1935.

[14] J.-Y. Girard. Linear Logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[15] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1989.

[16] T. Griffin. A Formulae-as-Types Notion of Control. In *Principles of Programming Languages*, pages 47–58. ACM Press, 1990.

[17] H. Herbelin. A λ-calculus Structure Isomorphic to Sequent Calculus Structure. In *Computer Science Logic*, volume 933 of *LNCS*, pages 67–75. Springer Verlag, 1995.

[18] J.-B. Joinet, H. Schellinx, and L. Tortora de Falco. SN and CR for Free-Style LK^{tq}: Linear Decorations and Simulation of Normalisation. Preprint No. 1067, Utrecht University, Department of Mathematics, 1998.

[19] S. C. Kleene. *Introduction to Metamathematics*. North-Holland, 1952.

[20] F. Pfenning. Structural Cut-Elimination. In *Logic and Computer Science*, pages 156–166. IEEE Computer Society, 1995.

[21] G. Pottinger. Normalisation as Homomorphic Image of Cut-Elimination. *Annals of Mathematical Logic*, 12:323–357, 1977.

[22] D. Prawitz. Ideas and Results of Proof Theory. In *Proceedings of the 2nd Scandinavian Logic Symposium*, volume 63 of *Studies in Logic and the Foundations of Mathematics*, pages 235–307. North-Holland, 1971.

[23] H. Schellinx. *The Noble Art of Linear Decorating*. PhD thesis, Institute for Logic, Language and Computation, University of Amsterdam, 1994. ILLC dissertation series.

[24] A. S. Troelstra and H. Schwichtenberg. *Basic Proof Theory*, volume 43 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1996.

[25] C. Urban. *Classical Logic and Computation*. PhD thesis, Cambridge University, October 2000.

[26] C. Urban and G. M. Bierman. Strong Normalisation of Cut-Elimination in Classical Logic. In *Typed Lambda Calculi and Applications*, volume 1581 of *LNCS*, pages 365–380. Springer Verlag, 1999.

[27] J. Zucker. The Correspondance Between Cut-Elimination and Normalisation. *Annals of Mathematical Logic*, 7:1–112, 1974.