Revisiting Zucker's Work on the Correspondence Between Cut-Elimination and Normalisation

Christian Urban

Abstract Zucker showed that in the fragment of intuitionistic logic whose formulae are build up from $\wedge$, $\supset$ and $\forall$ only, every reduction sequence in natural deduction corresponds to a reduction sequence in the sequent calculus and vice versa. Unfortunately, the technical machinery in Zucker's work is rather cumbersome and complicated. One contribution of this chapter is to greatly simplify his arguments. For example he defined a cut-elimination procedure modulo an equivalence relation; our cut-elimination procedure will be a simple term-rewriting system instead. Zucker also showed that the correspondence breaks down when the connectives $\vee$ or $\exists$ are included. We shall show that this negative result is not because cut-elimination fails to be strongly normalising for these connectives, as asserted by Zucker, rather it is because certain cut-elimination reductions do not correspond to any normalisation reduction.

Keywords Intuitionistic logic · Sequent calculus · Cut-elimination · Natural deduction · Normalisation

1 Introduction

Already Gentzen [7] presented a mapping from sequent proofs to natural deduction proofs, which however was later much improved by Prawitz [14]. According to this (many-to-one) mapping, instances of the cut-rule in the sequent calculus correspond to what is in natural deduction often called detours. From this observation arises the obvious question how cut-elimination, the process of eliminating all instances of the cut-rule from sequent proofs, and normalisation, the process of eliminating all detours from natural deduction proofs, correspond to each other under this mapping.

C. Urban (✉)
Department of Informatics, King's College London, London, UK
e-mail: christian.urban@kcl.ac.uk

L. C. Pereira et al. (eds.), *Advances in Natural Deduction*,
Trends in Logic 39, DOI: 10.1007/978-94-007-7548-0_2,

It seems Kreisel [10, p. 113] was the first who examined this "correspondence question" when he wrote:

> Consider…a calculus of sequents and a system of natural deduction. Inspection shows that …the normalization procedure for systems of natural deduction does not correspond to a particularly natural cut elimination procedure.

Indeed, if we employ Gentzen's method of eliminating cut-rules, that is eliminating innermost cut-rules first, then there is in general no correspondence with normalising natural deduction proofs. Consider for example the sequent proof

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{B \vdash B}\quad\dfrac{}{A \vdash A}}{B \supset A, B \vdash A}\supset_L}{B \supset A \vdash B \supset A}\supset_R \quad \dfrac{\dfrac{}{B \vdash B}\quad\dfrac{}{A \vdash A}}{B \supset A, B \vdash A}\supset_L}{B \supset A, B \vdash A}\ \mathrm{Cut}^{\bullet}}{B \vdash (B \supset A) \supset A}\supset_R \quad \dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}}{A, B \vdash A}\ \mathrm{Weak}_L}{A \vdash B \supset A}\supset_R \quad \dfrac{}{A \vdash A}}{(B \supset A) \supset A, A \vdash A}\supset_L}{A, B \vdash A}\ \mathrm{Cut}^{\star}$$

which maps to the following natural deduction proof.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{B \supset A \vdash B \supset A}\quad\dfrac{}{B \vdash B}}{B \supset A, B \vdash A}\supset_E}{B \supset A \vdash B \supset A}\supset_I^{\bullet} \quad \dfrac{}{B \vdash B}}{B, B \supset A \vdash A}\supset_E^{\bullet}}{B \vdash (B \supset A) \supset A}\supset_I^{\star} \quad \dfrac{\dfrac{}{A, B \vdash A}}{A \vdash B \supset A}\supset_I}{A, B \vdash A}\supset_E^{\star}$$

Note that the cut marked with a star maps to the starred detour (similarly the cut marked with a disc). Clearly we can remove the *-detour using the reductions introduced in Prawitz [14]. In order to obtain a corresponding cut-elimination sequence, we have to reduce the *-cut and move up the resulting smaller cuts, which is however *not* permitted by Gentzen's innermost strategy.

What Zucker [19] did is to remove the strategy and to introduce commuting conversions so that cuts can be eliminated independently from other cuts. A crucial point however is that cut-elimination needs to be strongly normalising, otherwise the correspondence fails: because Prawitz's reduction rules for natural deduction are strongly normalising, infinite cut-reduction sequences cannot be mapped to finite normalisation sequences. Under these circumstances, obtaining a strongly normalising cut-elimination procedure is rather tricky. As seen in the example above, the cut-elimination procedure has to allow cut-rules to pass over other cut-rules. For this consider the following cut-reduction, which allows a cut-rule (Suffix 2) to pass over another cut-rule (Suffix 1).

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

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. Zucker found a (rather *ad hoc*) way round this problem of loops: he considered only *proper reduction sequences*, that is reduction sequences which do not include repetitions; see Zucker [19, p. 93]. This restriction ensures that there are no infinite cut-reduction sequences in the $(\wedge, \supset, \forall)$-fragment of intuitionistic logic. In effect, he could give a positive answer to the correspondence question for this fragment (he also considered $\bot$, which we however shall omit). He established the following two facts ($|\_\_|$ stands for Prawitz's translation from sequent proofs to natural deduction proofs).

(i) Given a sequent proof M that reduces to N in one step. Then there is a (possibly empty) sequence of reductions starting from the natural deduction proof $|M|$ and ending with $|N|$.
(ii) Given a sequent proof M. If the natural deduction proof $|M|$ reduces to L, then there exist a sequent proof N such that $|N|$ is L and M reduces in one or more steps to N.

With these facts in place, Zucker could show that each reduction sequence in one system can be mapped to a reduction sequence in the other.

The restriction about proper, or non-repeating, reduction sequences fails, however, to ensure termination of cut-elimination when the connectives $\vee$ or $\exists$ are included. Figure 1 shows a simplified version of the non-terminating, non-repeating reduction sequence given by Zucker. Consequently, he gave a negative answer to the correspondence question for fragments that include $\vee$ or $\exists$. He wrote [19, p. 97]:

> We consider…a (natural) conversion rule in $\mathcal{S}$ for permuting cut with $\vee$L, which does not correspond to identity (or even a reduction) in $\mathcal{N}$. In fact, by use of this conversion rule, it turns out that one can define a non-termination, non-repeating reduction sequence in $\mathcal{S}$. It follows that for *any* set of conversion rules for $\mathcal{N}$ (with $\vee$) for which strong normalisation holds…the correspondence…between reductions in the two systems must fail.

In this paragraph $\mathcal{S}$ and $\mathcal{N}$ stand for Zucker's sequent calculus and natural deduction, respectively.

The main difference between Zucker's work and ours is that our cut-elimination procedure *is* strongly normalising for all connectives. Therefore we shall tackle the correspondence question again. In doing so we shall take care that the technical machinery is simpler than Zucker's. For example we shall not make use of an equivalence relation, which in his work includes Kleene-like permutation rules for cuts as well as rules for permuting contractions downwards in a sequent proof. Instead, we shall use the term-rewriting system introduced in Urban [17]; for a shorter account of this work see Urban and Bierman [18].

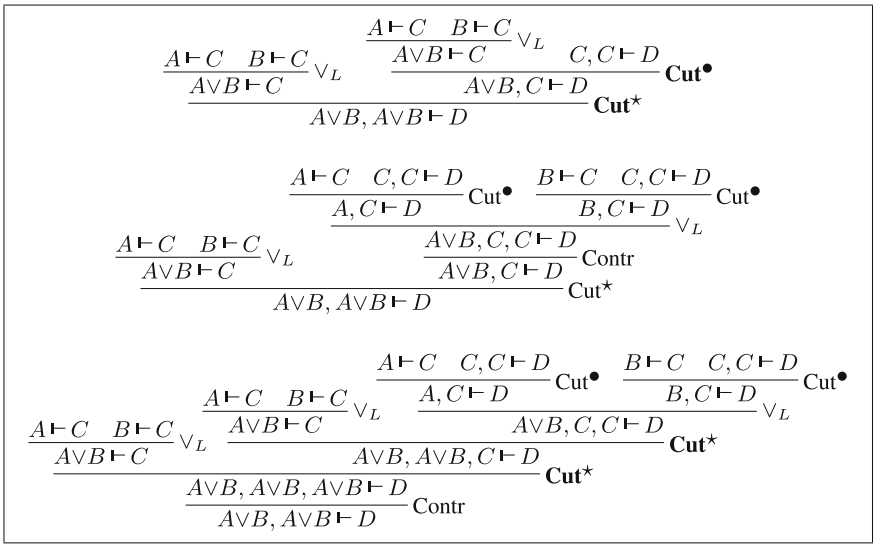

Fig. 1 A slight variant of the non-terminating reduction sequence given by Zucker; see Ungar [16]. In the first step, the cut marked with a disc is reduced creating two cuts as well as introducing a contraction. In the second step, the cut marked with a star is permuted with the contraction rule—again creating two cuts. Now the cuts written in bold face have the same shape as in the first proof, and we can repeat the reduction sequence infinitely many times

While the correspondence question seems to be of purely 'syntactic' interest, there are in fact a number of interesting issues. Natural deduction is the system of choice for any semantical investigation; to quote Girard et al. [8, p. 39]: 'In some sense, we should think of natural deductions as the true "proof" objects.' This is because in natural deduction (at least in some fragments) one can define a notion of equality of proof that is preserved under reduction. Consider now the correspondence question again: if cut-reductions do not match with normalisation reductions, then the sequent calculus might contain features that are not accounted for in natural deduction. This is particularly interesting, because in classical logic only sequent calculi seem to provide a framework for studying non-deterministic features; see Barbanera and Berardi [2], Urban [17] and Laird [11].

There are a number of earlier works that give an entirely positive answer to the correspondence question for all connectives. For example Pottinger [13] solved Zucker's problem with non-terminating reduction sequences by introducing normalisation-like reductions for the sequent calculus. Unfortunately, this work has a subtle defect: because Pottinger annotated sequent proofs with lambda-terms and then formalised cut-elimination as normalisation, the notion of normality for lambda-terms does *not* coincide with the notion of cut-freedom; see Pottinger [13, p. 323]. We shall avoid this problem by using proof annotations that encode precisely the structure of sequent proofs and also implement precisely the (standard) rules for cut-elimination.

More recently, Negri and von Plato [12] describe a natural deduction system with a particular formulation for the elimination rules—they are called general elimination rules. For this system a positive answer to the correspondence question can be given. However, we find it is questionable to make the natural deduction calculus to be 'sequent-calculus-like', meaning that many proofs are distinguished that differ only in the order of some inference rules. There is also work by Ungar [16] that modifies the natural deduction calculus for obtaining a positive answer to the correspondence question. His results build upon a very complicated, graph-like notion of natural deduction proofs, which is also not very convincing. Therefore, we feel it is prudent to reconsider the correspondence question.

The chapter is organised as follows: In Sects. 2 and 3 we shall describe a natural deduction system and a sequent calculus. In Sect. 4 we shall address the correspondence question for the $(\wedge, \supset, \forall)$-fragment of intuitionistic logic. An example will be given for why the correspondence breaks when $\vee$ is included. We shall conclude and suggest further work in Sect. 5.

2 Natural Deduction

For studying the correspondence question, Zucker introduced in [19] a minor variant of Gentzen's natural deduction system. We shall also use a slight variant, namely the system presented in (for example) Gallier [6]. It differs from Gentzen's formulation of natural deduction in that proofs are written in sequent style—that means open assumptions are recorded explicitly in every inference step—and that terms are annotated to proofs.

First we give the grammars for expressions and formulae (we use 'expression' instead of 'term' and moreover write expressions in a sans serif font in order to avoid confusions with proof annotations we shall introduce later).

$$\begin{aligned} \mathsf{t} &::= \mathsf{x} \mid \mathsf{f}\,\mathsf{t}_1 \ldots \mathsf{t}_n \\ B &::= A\,\mathsf{t}_1 \ldots \mathsf{t}_n \mid B \wedge B \mid B \vee B \mid B \supset B \mid \forall \mathsf{x}.B \mid \exists \mathsf{x}.B \end{aligned}$$

In this grammar x is taken from a set of variables, f from a set of functional symbols and A ranges over predicate symbols. As usual functional and predicate symbols have an arity (a natural number) assigned specifying the number of arguments. We shall often write x, y, z for variables and t for arbitrary expressions; square brackets (e.g., $[\mathsf{x}]$) will indicate that a variable becomes bound; and capture avoiding substitution for expressions and formulae is assumed as usual.

Next we define lambda-terms for annotating natural deduction proofs. Raw lambda-terms are defined by the grammar

p

$$
\begin{array}{lll}
M, N, P ::= x & & \text{Var} \\
\quad | \quad \langle M, N \rangle & & \text{And-I} \\
\quad | \quad \mathtt{fst}(M) \mid \mathtt{snd}(M) & & \text{And-E}_i \quad i = 1, 2 \\
\quad | \quad \mathtt{inl}(M) \mid \mathtt{inr}(M) & & \text{Or-I}_i \quad i = 1, 2 \\
\quad | \quad \mathtt{case}(P, \lambda x : B'.M, \lambda y : B''.N) & & \text{Or-E} \\
\quad | \quad \lambda x : B.M & & \text{Imp-I} \\
\quad | \quad M\ N & & \text{Imp-E} \\
\quad | \quad \mathtt{inx}(\mathsf{t}, M) & & \text{Exists-I} \\
\quad | \quad \mathtt{casex}(M, \lambda x{:}B.[\mathsf{y}]N) & & \text{Exists-E} \\
\quad | \quad [\mathsf{y}]M & & \text{Forall-I} \\
\quad | \quad M\,\mathsf{t} & & \text{Forall-E}
\end{array}
$$

where y and t are expressions, and the Bs are formulae. Let us briefly mention the conventions we shall assume for lambda-terms: free and bound variables are as usual; for brevity we shall often omit the formulae on binders and simply write $\lambda x.M$; a Barendregt-style naming convention for variables will always be observed and lambda-terms will be regarded as equivalent up to renaming of bound variables. A pleasing consequence of these conventions is that the notion of variable substitution for lambda-terms can be defined without worrying about clashes and capture of variables.

To annotate lambda-terms to natural deduction proofs, we shall have sequents of the form $\Gamma \triangleright M : B$ in which M is a lambda-term, B is a formula and Γ is a context—a set of (variable, formula)-pairs. We shall also employ some shorthand notation for contexts: rather than writing for example $\{(x, B), (y, C), (z, D)\}$, we shall write $x : B, y : C, z : D$. For contexts we have the convention that a context is *ill-formed*, if it contains more than one occurrence of a variable. For example, the context $x : B, x : C$ is not allowed.

In the sequel we are only interested in lambda-terms, M, for which there is a context Γ and a formula B such that $\Gamma \triangleright M : B$ is derivable given the inference rules shown in Fig. 2. The following set is defined to contain all those lambda-terms.

$$\mathcal{N} \stackrel{\text{def}}{=} \{\ M \mid \Gamma \triangleright M : B \text{ is derivable using the rules given in Fig. 2}\ \}$$

Apart from the initial sequent, the inference rules shown in Fig. 2 fall into two groups: introduction and elimination rules. A *detour* is a sequence of neighbouring inference rules beginning with an introduction rule introducing a formula that is eliminated by the last one in the sequence (necessarily an elimination rule). Detours in natural deduction proofs violate the *subformula property*; for a definition of this standard property see Troelstra and Schwichtenberg [15]. Normalisation is the process of stepwise eliminating detours from natural deduction proofs, converting them into normal forms for which the subformula property holds. Below we shall introduce term-rewriting rules, commonly referred to as *beta-reductions*, which eliminate a detour formed by an introduction rule and a neighbouring elimination rule.

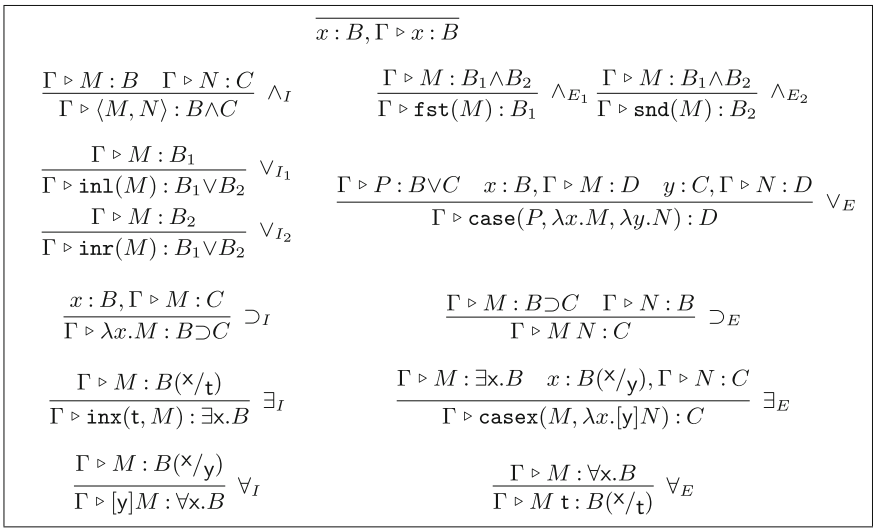

Fig. 2 Term assignment for intuitionistic natural deduction proofs

$$
\begin{aligned}
\mathtt{fst}(\langle M, N\rangle) &\xrightarrow{\beta} M\\
\mathtt{snd}(\langle M, N\rangle) &\xrightarrow{\beta} N\\
\mathtt{case}(\mathtt{inl}(P), \lambda x.M, \lambda y.N) &\xrightarrow{\beta} M(x/P)\\
\mathtt{case}(\mathtt{inr}(P), \lambda x.M, \lambda y.N) &\xrightarrow{\beta} N(y/P)\\
(\lambda x.M)\, N &\xrightarrow{\beta} M(x/N)\\
\mathtt{casex}(\mathtt{inx}(\mathsf{t}, M), \lambda x.[\mathsf{y}]N) &\xrightarrow{\beta} N(\mathsf{y}/t)(\mathsf{x}/M)\\
([\mathsf{y}]M)\,\mathsf{t} &\xrightarrow{\beta} M(\mathsf{y}/\mathsf{t})
\end{aligned}
$$

In order to ensure that the normal forms satisfy the subformula property, further rewriting rules (we shall call them *gamma-reductions*) are needed for the term-constructors `case` and `casex`. A lucid explanation for why further rules are necessary is given in Girard et al. [8, p. 74]. We shall present the rewriting rules for `case` only (the rules for `casex` are similar).

$$
\begin{aligned}
\mathtt{fst}(\mathtt{case}(P, \lambda x.M, \lambda y.N)) &\xrightarrow{\gamma} \mathtt{case}(P, \lambda x.\mathtt{fst}(M), \lambda y.\mathtt{fst}(N))\\
\mathtt{snd}(\mathtt{case}(P, \lambda x.M, \lambda y.N)) &\xrightarrow{\gamma} \mathtt{case}(P, \lambda x.\mathtt{snd}(M), \lambda y.\mathtt{snd}(N))\\
\mathtt{case}(P, \lambda x.M, \lambda y.N)\, Q &\xrightarrow{\gamma} \mathtt{case}(P, \lambda x.(M\, Q), \lambda y.(N\, Q))\\
\mathtt{case}(P, \lambda x.M, \lambda y.N)\,\mathsf{t} &\xrightarrow{\gamma} \mathtt{case}(P, \lambda x.(M\,\mathsf{t}), \lambda y.(N\,\mathsf{t}))\\
\mathtt{case}(\mathtt{case}(P, \lambda x.M, \lambda y.N), \lambda u.S, \lambda v.T) &\xrightarrow{\gamma} \mathtt{case}(P, \lambda x.\mathtt{case}(M, \lambda u.S, \lambda v.T),\\
&\qquad \lambda y.\mathtt{case}(N, \lambda u.S, \lambda v.T))\\
\mathtt{casex}(\mathtt{case}(P, \lambda x.M, \lambda y.N), \lambda z.[\mathsf{y}]S) &\xrightarrow{\gamma} \mathtt{case}(P, \lambda x.\mathtt{casex}(M, \lambda z.[\mathsf{y}]S),\\
&\qquad \lambda y.\mathtt{casex}(N, \lambda z.[\mathsf{y}]S))
\end{aligned}
$$

We automatically assume that the reduction rules are closed under context formation, which is a standard convention in term rewriting. Properties of these reduction rules such as subject reduction, strong normalisation and confluence are well-known and therefore not further elaborated here.

3 Sequent Calculus

Zucker's chapter is not very concise, in part because he formalises a sequent calculus using the rather cumbersome notion of indexed formulae. A simpler formulation for an intuitionistic sequent calculus is given in Urban [17]; for an overviw of this work see Urban and Bierman [18]. The main idea is to formalise the sequent calculus in a similar fashion as natural deduction using proof-annotations. This will mean for example that contexts are sets, as in type-theory, and *not* multisets, as in Gentzen's LJ. A pleasing feature of 'contexts-as-sets' is that the structural rules are completely implicit in the form of the logical rules. In the remainder of this section, we shall repeat the basic definitions from Urban [17] in order to make this chapter self-contained.

Formulae and expressions are defined as in natural deduction. Sequents are of the form

$$\Gamma \vartriangleright M \vartriangleright a : B$$

where Γ is a context, $a : B$ is a labelled formula and M is a term. As in Sect. 2, contexts are sets of labelled formulae. However, in the sequent calculus we shall call the labels *names*; so contexts are sets of (name,formula)-pairs. Otherwise the same conventions for contexts apply as given in Sect. 2. The formula on the right-hand side of the sequent is labelled by a *co-name*. While co-names can be dispensed with when formalising an intuitionistic sequent calculus (they are necessary for classical logic), we shall *not* suppress them for reasons that will become clear later on when we define the operation of proof substitution.

The raw terms for annotating sequent proofs are given by the grammar:

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

in which x, y, z are taken from a set of names; a, b, c from a set of co-names; y and t are expressions; and the Bs are formulae. We shall often write …,x, y, z for names, and a, b, c, …for co-names. Round brackets signify that a name becomes bound and angle brackets, that a co-name becomes bound. Again we have similar conventions as in Sect. 2: we shall omit the types on the bindings; regard terms as equal up to alpha-conversions and adopt a Barendregt-style convention for 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 lambda-terms: whilst a variable can be substituted with a lambda-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 the terms defined above 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 denoted by $FN(M)$ and $FC(M)$, respectively. Another useful notion is the following.

Definition 1 A term, M, introduces the name z or co-name c, if and only if M is of the form

for z:	for c:
$\mathsf{Ax}(z, c)$	$\mathsf{Ax}(z, c)$
$\mathsf{And}^i_L((x)S, z)$	$\mathsf{And}_R(\langle a\rangle S, \langle b\rangle T, c)$
$\mathsf{Or}_L((x)S, (y)T, z)$	$\mathsf{Or}^i_R(\langle a\rangle S, c)$
$\mathsf{Imp}_L(\langle a\rangle S, (x)T, z)$	$\mathsf{Imp}_R((x)\langle a\rangle S, c)$
$\mathsf{Exists}_L((x)[\mathsf{y}]S, z)$	$\mathsf{Exists}_R(\langle a\rangle S, \mathsf{t}, c)$
$\mathsf{Forall}_L((x)S, \mathsf{t}, z)$	$\mathsf{Forall}_R(\langle a\rangle[\mathsf{y}]S, c)$

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 a main formula of an inference.

Again we shall be interested in only *well-typed* terms; this means those M for which there is a context Γ and a labelled formula $a : B$, such that $\Gamma \triangleright M \triangleright a : B$ holds given the inference rules in Fig. 3. The following set contains all such terms.

$$\mathcal{S} \stackrel{\text{def}}{=} \{\, M \mid \Gamma \triangleright M \triangleright a : B \text{ is derivable using the rules given in Fig. 3} \,\}$$

Whilst the structural rules are *implicit* in our sequent calculus, i.e. the calculus has fewer inference rules, there are a number of subtleties concerning contexts. We assume for the commas in Fig. 3 the following conventions: a comma in a conclusion stands for set union and a comma in a premise stands for *disjoint* set union. Consider for example the $\wedge_{L_i}$-rule. This rule introduces the (name,formula)-pair $y : B_1 \wedge B_2$ in the conclusion, and consequently, y is a free name in $\mathsf{And}^i_L((x)M, y)$. However, y can already be free in the subterm M, in which case $y : B_1 \wedge B_2$ belongs to Γ. We refer to this as an *implicit contraction*. Hence the left-hand side of the conclusion of

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

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

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

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

$$\frac{x : B({}^{\mathsf{x}}/_{\mathsf{y}}), \Gamma \triangleright M \triangleright a : C}{y : \exists \mathsf{x}.B, \Gamma \triangleright \mathsf{Exists}_L((x)[\mathsf{y}]M, y) \triangleright a : C}\,\exists_L \qquad \frac{\Gamma \triangleright M \triangleright a : B({}^{\mathsf{x}}/_{\mathsf{t}})}{\Gamma \triangleright \mathsf{Exists}_R(\langle a \rangle M, \mathsf{t}, b) \triangleright b : \exists \mathsf{x}.B}\,\exists_R$$

$$\frac{x : B({}^{\mathsf{x}}/_{\mathsf{t}}), \Gamma \triangleright M \triangleright a : C}{y : \forall \mathsf{x}.B, \Gamma \triangleright \mathsf{Forall}_L((x)M, \mathsf{t}, y) \triangleright a : C}\,\forall_L \qquad \frac{\Gamma \triangleright M \triangleright a : B({}^{\mathsf{x}}/_{\mathsf{y}})}{\Gamma \triangleright \mathsf{Forall}_R(\langle a \rangle [\mathsf{y}]M, b) \triangleright b : \forall \mathsf{x}.B}\,\forall_R$$

Fig. 3 Term assignment for sequent proofs

$\wedge_{L_i}$ is of the form $y : B_1 \wedge B_2 \oplus \Gamma$ where $\oplus$ denotes set union. Clearly, in the case that the term $\mathsf{And}^i_L((x)M, y)$ freshly introduces y, then this context is of the form $y : B_1 \wedge B_2 \otimes \Gamma$ where $\otimes$ denotes *disjoint* set union. Note that $x : B_i$ cannot be part of the conclusion: x is intended to become bound in the term. Thus the context in the premise must be of the form $x : B_i \otimes \Gamma$.

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

$$\frac{\Gamma_1 \triangleright M \triangleright a : B \quad x : B \otimes \Gamma_2 \triangleright N \triangleright b : C}{\Gamma_1 \oplus \Gamma_2 \triangleright \mathsf{Cut}(\langle a \rangle M, (x)N) \triangleright b : C}\,\mathrm{Cut}$$

In effect, this rule is only applicable, if it does not break the convention about ill-formed contexts, which can always be achieved by renaming some names appropriately. Notice however that we do not require that cut-rules have to be "fully" multiplicative: the Γ_is can share some formulae.

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 LJ for example there are 360 different cases of commuting cuts! 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 24 clauses. Clearly, this is an advantage of our use of terms.

Before we give the definition of the substitution, it is instructive to look at some examples. Commuting cuts need to permute to the places where the cut-formula is the

main formula. At the level of terms this means the cuts need to be permuted to every subterm that introduces the cut-formula. This will be achieved with substitutions of the form

$$P(x{:}B/\langle a{:}B\rangle Q) \quad \text{and} \quad S(b{:}B/(y{:}B)T).$$

If 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 (we do not write the type annotations in substitutions provided they are clear from the context).

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

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 the second case, the substitution is also pushed inside the subterms. This is because there might be several occurrences of x (this name need not have been freshly introduced). An exception applies to axioms, where the substitution is defined differently, as shown below.

$$\mathsf{Ax}(x, a)(x/\langle b\rangle P) = P[b \mapsto a]$$
$$\mathsf{Ax}(x, a)(a/(y)Q) = Q[y \mapsto x]$$

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.

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

Figure 4 gives the complete definition of proof 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.

We are now in the position to formalise our cut-elimination procedure. We shall distinguish two kinds of cuts: *logical cuts* and *commuting cuts*. A cut of the form $\mathsf{Cut}(\langle a\rangle M, (x)N)$ is a logical cut provided M freshly introduces a and N freshly introduces x; otherwise the cut is a commuting cut. The corresponding reduction rules

$$
\begin{array}{rrcl}
1. & \mathsf{Ax}(x,c)(c/_{\langle y\rangle}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{And}_R(\langle a\rangle M,\langle b\rangle N,c)(c/_{\langle y\rangle}P) & \stackrel{\text{def}}{=} & \mathsf{Cut}(\langle c\rangle\mathsf{And}_R(\langle a\rangle M,\langle b\rangle N,c),\langle y\rangle P)\\
4. & \mathsf{Or}^i_R(\langle a\rangle M,c)(c/_{\langle y\rangle}P) & \stackrel{\text{def}}{=} & \mathsf{Cut}(\langle c\rangle\mathsf{Or}^i_R(\langle a\rangle M,c),\langle y\rangle P)\\
5. & \mathsf{Imp}_R((x)\langle a\rangle M,b)(b/_{\langle y\rangle}P) & \stackrel{\text{def}}{=} & \mathsf{Cut}(\langle b\rangle\mathsf{Imp}_R((x)\langle a\rangle M,b),\langle y\rangle P)\\
6. & \mathsf{Exists}_R(\langle a\rangle M,\mathsf{t},b)(b/_{\langle y\rangle}P) & \stackrel{\text{def}}{=} & \mathsf{Cut}(\langle b\rangle\mathsf{Exists}_R(\langle a\rangle M,\mathsf{t},b),\langle y\rangle P)\\
7. & \mathsf{Forall}_R(\langle x\rangle[\mathsf{y}]M,b)(b/_{\langle y\rangle}P) & \stackrel{\text{def}}{=} & \mathsf{Cut}(\langle b\rangle\mathsf{Forall}_R(\langle a\rangle[\mathsf{y}]M,b),\langle y\rangle P)\\
8. & \mathsf{And}^i_L((x)M,y)(y/_{\langle c\rangle}P) & \stackrel{\text{def}}{=} & \mathsf{Cut}(\langle c\rangle P,\langle y\rangle\mathsf{And}^i_L((x)M(y/_{\langle c\rangle}P),y))\\
9. & \mathsf{Or}_L((x)M,\langle y\rangle N,z)(z/_{\langle c\rangle}P) & \stackrel{\text{def}}{=} & \\
 & & & \mathsf{Cut}(\langle c\rangle P,\langle z\rangle\mathsf{Or}_L((x)M(z/_{\langle c\rangle}P),\langle y\rangle N(z/_{\langle c\rangle}P),z))\\
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,\langle y\rangle\mathsf{Imp}_L(\langle a\rangle M(y/_{\langle c\rangle}P),(x)N(y/_{\langle c\rangle}P),y))\\
11. & \mathsf{Exists}_L((x)[\mathsf{y}]M,y)(y/_{\langle c\rangle}P) & \stackrel{\text{def}}{=} & \mathsf{Cut}(\langle c\rangle P,\langle y\rangle\mathsf{Exists}_L((x)[\mathsf{y}]M(y/_{\langle c\rangle}P),y))\\
12. & \mathsf{Forall}_L((x)M,\mathsf{t},y)(y/_{\langle c\rangle}P) & \stackrel{\text{def}}{=} & \mathsf{Cut}(\langle c\rangle P,\langle y\rangle\mathsf{Forall}_L((x)M(y/_{\langle c\rangle}P),\mathsf{t},y))
\end{array}
$$

Otherwise:

$$
\begin{array}{rrcl}
13. & \mathsf{Ax}(x,a)[\sigma] & \stackrel{\text{def}}{=} & \mathsf{Ax}(x,a)\\
14. & \mathsf{Cut}(\langle a\rangle M,(x)N)[\sigma] & \stackrel{\text{def}}{=} & \mathsf{Cut}(\langle a\rangle\ M[\sigma],(x)\ N[\sigma])\\
15. & \mathsf{And}_R(\langle a\rangle M,\langle b\rangle N,c)[\sigma] & \stackrel{\text{def}}{=} & \mathsf{And}_R(\langle a\rangle\ M[\sigma],\langle b\rangle\ N[\sigma],c)\\
16. & \mathsf{And}^i_L((x)M,y)[\sigma] & \stackrel{\text{def}}{=} & \mathsf{And}^i_L((x)\ M[\sigma],y)\\
17. & \mathsf{Or}^i_R(\langle a\rangle M,b)[\sigma] & \stackrel{\text{def}}{=} & \mathsf{Or}^i_R(\langle a\rangle\ M[\sigma],b)\\
18. & \mathsf{Or}_L((x)M,\langle y\rangle N,z)[\sigma] & \stackrel{\text{def}}{=} & \mathsf{Or}_L((x)\ M[\sigma],\langle y\rangle\ N[\sigma],z)\\
19. & \mathsf{Imp}_R((x)\langle a\rangle M,b)[\sigma] & \stackrel{\text{def}}{=} & \mathsf{Imp}_R((x)\langle a\rangle\ M[\sigma],b)\\
20. & \mathsf{Imp}_L(\langle a\rangle M,(x)N,y)[\sigma] & \stackrel{\text{def}}{=} & \mathsf{Imp}_L(\langle a\rangle\ M[\sigma],(x)\ N[\sigma],y)\\
21. & \mathsf{Exists}_R(\langle a\rangle M,\mathsf{t},b)[\sigma] & \stackrel{\text{def}}{=} & \mathsf{Exists}_R(\langle a\rangle\ M[\sigma],\mathsf{t},b)\\
22. & \mathsf{Exists}_L((x)[\mathsf{y}]M,y)[\sigma] & \stackrel{\text{def}}{=} & \mathsf{Exists}_L((x)[\mathsf{y}]\ M[\sigma],y)\\
23. & \mathsf{Forall}_R(\langle a\rangle[\mathsf{y}]M,b)[\sigma] & \stackrel{\text{def}}{=} & \mathsf{Forall}_R(\langle a\rangle[\mathsf{y}]\ M[\sigma],b)\\
24. & \mathsf{Forall}_L((x)M,\mathsf{t},y)[\sigma] & \stackrel{\text{def}}{=} & \mathsf{Forall}_L((x)\ M[\sigma],\mathsf{t},y)
\end{array}
$$

Fig. 4 Proof substitution

are given in Fig. 5. The side-conditions attached to each rule ensure that commuting cuts cannot be rewritten using a rule for a logical cut. Again we automatically assume that $\stackrel{c}{\longrightarrow}$ and $\stackrel{l}{\longrightarrow}$ are closed under context formation. We shall write $\stackrel{cut}{\longrightarrow}$ whenever we refer to both $\stackrel{c}{\longrightarrow}$ and $\stackrel{l}{\longrightarrow}$.

Urban [17] and Urban and Bierman [18] present a detailed proof showing that $\stackrel{cut}{\longrightarrow}$ is strongly normalising (even in the classical case). However, $\stackrel{cut}{\longrightarrow}$ is *not* confluent!

Logical Cuts $(i = 1, 2)$

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

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

3. $\mathsf{Cut}(\langle b\rangle \mathsf{Imp}_R((x)\langle a\rangle M, b), (z)\mathsf{Imp}_L(\langle c\rangle N, (y)P, z))$
   $\xrightarrow{l} \mathsf{Cut}(\langle a\rangle \mathsf{Cut}(\langle c\rangle N, (x)M), (y)P)$ 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

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

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

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

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

Commuting Cuts

8. $\mathsf{Cut}(\langle a\rangle M, (x)N)$
   $\xrightarrow{c} M(^a/_{(x)N})$ if M does not freshly introduce a, or
   $\xrightarrow{c} N(^x/_{\langle a\rangle M})$ if N does not freshly introduce x

Fig. 5 Cut-reductions for logical and commuting cuts

4 The Correspondence Question

Before we can answer the correspondence question, we need to formalise the translation from sequent proofs to natural deduction proofs. Figure 6 shows the standard translation, which appeared first in Prawitz [14]. It translates inductively a sequent proof so that right-rules are mapped to introduction rules on the root of natural deduction proofs, and left-rules to elimination rules at the top (except $\vee_L$ and $\exists_L$ which translate like right-rules). It is not hard to show that this translation is onto, but a proof is omitted. First we show that typed terms translate to typed terms.

Proposition 2 *For all* $M \in \mathcal{S}$*, if the sequent* $\Gamma \triangleright M \triangleright a : B$ *is derivable, then* $\Gamma \triangleright |M| : B$ *is derivable and vice versa.*

Proof Routine induction on the structure of M. □

Next we shall show how the symmetric operation of proof-substitution relates to the usual lambda-term substitution. We shall first consider the $(\wedge, \supset, \forall)$-fragment.

$$
\begin{array}{rcl@{\qquad}rcl}
|\mathsf{Ax}(x,a)| & \stackrel{\text{def}}{=} & x & |\mathsf{Cut}(\langle a\rangle M, (x)N)| & \stackrel{\text{def}}{=} & |N|(^{x}/_{|M|}) \\
|\mathsf{And}_R(\langle a\rangle M, \langle b\rangle N, c)| & \stackrel{\text{def}}{=} & \langle |M|, |N|\rangle & |\mathsf{And}^i_L((x)M, y)| & \stackrel{\text{def}}{=} & \begin{cases} |M|(^{x}/_{\mathtt{fst}(y)}) \\ |M|(^{x}/_{\mathtt{snd}(y)}) \end{cases} \\
|\mathsf{Or}^i_R(\langle a\rangle M, b)| & \stackrel{\text{def}}{=} & \begin{cases} \mathtt{inr}(|M|) \\ \mathtt{inl}(|M|) \end{cases} & |\mathsf{Or}_L((x)M, (y)N, z)| & \stackrel{\text{def}}{=} & \mathtt{case}(z, \lambda x.|M|, \lambda y.|N|) \\
|\mathsf{Imp}_R((x)\langle a\rangle M, c)| & \stackrel{\text{def}}{=} & \lambda x.|M| & |\mathsf{Imp}_L(\langle a\rangle M, (x)N, y)| & \stackrel{\text{def}}{=} & |N|(^{x}/_{(y\ |M|)}) \\
|\mathsf{Exists}_R(\langle a\rangle M, \mathsf{t}, b)| & \stackrel{\text{def}}{=} & \mathtt{inx}(\mathsf{t}, |M|) & |\mathsf{Exists}_L((x)[\mathsf{y}]M, y)| & \stackrel{\text{def}}{=} & \mathtt{casex}(y, \lambda x.[\mathsf{y}]|M|) \\
|\mathsf{Forall}_R(\langle a\rangle[\mathsf{y}]M, b)| & \stackrel{\text{def}}{=} & [\mathsf{y}]|M| & |\mathsf{Forall}_L((x)M, \mathsf{t}, y)| & \stackrel{\text{def}}{=} & |M|(^{x}/_{y\,\mathsf{t}})
\end{array}
$$

Fig. 6 Translation from sequent proofs to natural deduction proofs

Lemma 3 *For all* $M, N \in \mathcal{S}^{(\wedge,\supset,\forall)}$ *we have*

(i) $|M(x/\langle a\rangle N)| \equiv |M|[x/|N|]$
(ii) $|M(a/(x)N)| \equiv |N|[x/|M|]$ *provided* $a \in FC(M)$.

Proof By induction on the structure of M. □

Note that the side-condition in (ii) is always satisfied in intuitionistic logic for proof-substitutions which arise from reducing commuting cuts. This lemma enables us to show that every cut-elimination reduction maps to zero or more normalisation reductions.

Theorem 4 *For all* $M, N \in \mathcal{S}^{(\wedge,\supset,\forall)}$, *if* $M \xrightarrow{cut} N$, *then* $|M| \xrightarrow{\beta} {}^{*}|N|$.

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

In particular we have that if $M \xrightarrow{c} N$ then $|M| \equiv |N|$. This fact will be useful for proving the following lemma, which relates normalisation steps to a sequence of cut-elimination steps.

Lemma 5 *Given an* $M \in \mathcal{S}^{(\wedge,\supset,\forall)}$ *and* $|M| \xrightarrow{\beta} L$. *Then there exists an* $N \in \mathcal{S}^{(\wedge,\supset,\forall)}$ *such that* $M \xrightarrow{cut} {}^{+} N$ *and* $L \xrightarrow{\beta} {}^{*}|N|$.

Proof By induction on M, but we can restrict the induction to Ms which have only logical cuts. To see this apply as many $\xrightarrow{c}$-reductions as possible to M, obtaining a $\xrightarrow{c}$-normal form, say M'. We have that $M \xrightarrow{c} {}^{*} M'$ and $|M| \equiv |M'|$. Before we analyse one case, some useful terminology will be introduced. Let us write $P\{x_1, \ldots, x_n\}$ for the lambda-term P having n free occurrences of the variable x, and thus $P\{Q_1, \ldots, Q_n\}$ will stand for $P[x/Q]$. Notice that if the outermost term-constructor of Q does not correspond to an introduction rule, then a reduction in $P\{Q_1, \ldots, Q_n\}$ can occur only in the 'P-part' or in one of the 'Q_i-parts'. With this fact in place, let us consider a $\wedge_R/\wedge_{L_1}$-logical cut having the form $\mathsf{Cut}(\langle c\rangle \mathsf{And}_R(\langle a\rangle S, \langle b\rangle T, c), (y)\mathsf{And}^1_L((x)R, y))$. Then $|M| =$

$|R|[x/\mathtt{fst}(\langle|S|, |T|\rangle)]$ which we can write equivalently as $|R|\{\mathtt{fst}\ (\langle|S|, |T|\rangle)_1, \ldots, \mathtt{fst}(\langle|S|, |T|\rangle)_n\}$. There are three cases to be dealt with for a reduction occurring in $|M|$. First, if the reduction occurs in the $|R|$-part, say $|R| \xrightarrow{\beta} |R'|$, then we can conclude by appealing to the induction hypothesis. In this case N is M except R is replaced by R'. Second, if the reduction occurs in one of the subterms of $\langle|S|, |T|\rangle$, say $|S| \xrightarrow{\beta} |S'|$ (the other subcase being similar), then we have the reduction $|M| \xrightarrow{\beta} |R|\{\ldots, \mathtt{fst}(\langle|S'|, |T|\rangle)_i, \ldots\} \xrightarrow{\beta}{}^* |R|\{\mathtt{fst}(\langle|S'|, |T|\rangle)_1, \ldots, \mathtt{fst}(\langle|S'|, |T|\ \rangle)_n\}$. The last term is the term where all other 'copies' of $|S|$ have been reduced, too. Again we can appeal to the induction hypothesis and conclude with N being M except S is replaced by S'. In the third case one of the $\mathtt{fst}(\langle|S|, |T|\rangle)$ reduces to $|S|$. Hence, we have the reduction $|M| \xrightarrow{\beta} |R|\{\ldots, |S|_i, \ldots\} \xrightarrow{\beta}{}^* |R|\{|S|_1, \ldots, |S|_n\}$. In this case we can conclude with $N \equiv \mathsf{Cut}(\langle a\rangle S, (x)R)$ and $M \xrightarrow{c}{}^* M' \xrightarrow{l} N$ as required. The other logical cuts are treated similarly. □

Unfortunately, this is a slightly weaker correspondence as obtained by Zucker [19, p. 79, Theorem 2]. This is because of the way the translation $|\_\_|$ is set up and also because we are not permuting contraction rules downwards in a sequent proof as Zucker does. In effect, it is *not* true that every normalisation step is matched by a series of cut-elimination steps, rather we have to consider 'clusters' of normalisation steps. So in order to obtain a result concerning the correspondence question we prove the following theorem.

Theorem 6 *Given an* $M \in \mathcal{S}^{(\wedge,\supset,\forall)}$ *and* $|M| \xrightarrow{\beta}{}^* L_{\mathtt{nf}}$ *where* $L_{\mathtt{nf}}$ *is a normal form. Then there exists an* $N \in \mathcal{S}^{(\wedge,\supset,\forall)}$ *such that* $M \xrightarrow{cut}{}^* N$ *and* $|N| = L_{\mathtt{nf}}$.

Proof We can use the fact that $\xrightarrow{\beta}$ is confluent and strongly normalising. The induction is on the length of the longest reduction sequence starting from $|M|$. By confluence we can repeatedly apply Lemma 5 until we reach the normal form $L_{\mathtt{nf}}$ of $|M|$. In this way, we construct the reduction sequence $M \xrightarrow{cut}{}^* N$. □

In effect we have given a positive answer to the correspondence question in the $(\wedge, \supset, \forall)$-fragment: every cut-reduction sequence maps to a beta-reduction sequence, and every beta-reduction sequence leading to a normal form can be mapped to a cut-reduction sequence. Let us now reflect on this answer and see whether the result Theorem 6 can be improved and whether the answer can be extended to $\vee$ and $\exists$.

It has been suggested for example by Barendregt and Ghilezan [3], Herbelin [9] and Espírito Santo [5] that lambda-calculi extended with explicit substitution operators provide a better correspondence result, meaning that every beta-reduction can be matched by a series of cut-elimination steps. As can be seen in Lemma 5 a cut, say $\mathsf{Cut}(\langle a\rangle M, (x)N)$, is translated as $|N|\{|M|_1, \ldots, |M|_n\}$. Now if any of the $|M|_i$ does a reduction, then this cannot be 'matched' in the cut. Instead if we translate the cut using an explicit substitution operator, for example $|N|\langle x := |M|\rangle$, we can avoid this problem. However we are not aware any work that points out that with

explicit substitution calculi Theorem 4 fails—already in the $(\supset)$-fragment! We shall demonstrate this phenomenon within the explicit substitution calculus λx; see for example Bloo [4].

Example 7 The terms of λx are given by the grammar

$$x \mid \lambda x.M \mid M\,N \mid M\langle x := N\rangle\,.$$

In contrast to the lambda calculus, in λx beta-reductions are split into more atomic steps: one step is the reduction $(\lambda x.M)N \xrightarrow{\beta'} M\langle x := N\rangle$ introducing the explicit substitution operator, and the others involve reductions that permute this operator inside the term M. As a consequence, the translation from sequent proofs to natural deduction proofs can be modified as follows.

$$\begin{aligned}
|\mathsf{Cut}(\langle a\rangle M, (x)N)|' &\stackrel{\text{def}}{=} |N|'\langle x := |M|'\rangle\\
|\mathsf{Imp}_R((x)\langle a\rangle M, c)|' &\stackrel{\text{def}}{=} \lambda x.|M|'\\
|\mathsf{Imp}_L(\langle a\rangle M, (x)N, y)|' &\stackrel{\text{def}}{=} |N|'\langle x := (y\ |M|')\rangle
\end{aligned}$$

To show that the correspondence with cut-elimination fails, consider the logical cut-reduction for $\supset_R/\supset_L$. In Fig. 3 we defined this cut-reduction as

$$\begin{aligned}
&\mathsf{Cut}(\langle b\rangle\mathsf{Imp}_R((x)\langle a\rangle M, b), (z)\mathsf{Imp}_L(\langle c\rangle N, (y)P, z))\\
&\quad\xrightarrow{l}\mathsf{Cut}(\langle a\rangle\mathsf{Cut}(\langle c\rangle N, (x)M), (y)P)\ \ \textbf{or}\\
&\quad\xrightarrow{l}\mathsf{Cut}(\langle c\rangle N, (x)\mathsf{Cut}(\langle a\rangle M, (y)P))
\end{aligned}\tag{1}$$

including a choice of how the cuts are arranged in the reduct (there is no reason to prefer one choice over the other). Using the translation $|\_\_|'$ given above only the first reduction is matched by reductions in λx; the other is *not*. The translations of the reducts in (1) are:

$$\begin{aligned}
|\mathsf{Cut}(\langle a\rangle\mathsf{Cut}(\langle c\rangle N, (x)M), (y)P)|' &= |P|'\langle y := |M|'\langle x := |N|'\rangle\rangle\\
|\mathsf{Cut}(\langle c\rangle N, (x)\mathsf{Cut}(\langle a\rangle M, (y)P))|' &= |P|'\langle y := |M|'\rangle\langle x := |N|'\rangle
\end{aligned}$$

With the (standard) reduction rules of λx, only the first reduct can be reached.

$$
\begin{array}{ll}
|\mathsf{Cut}(\langle b\rangle \mathsf{Imp}_R((x)\langle a\rangle M, b), (z)\mathsf{Imp}_L(\langle c\rangle N, (y)P, z))|' & \\
= & |P|'\langle y := z\,|N|'\rangle\langle z := \lambda x.|M|'\rangle \\
\longrightarrow & |P|'\langle y := z\,|N|'\langle z := \lambda x.|M|'\rangle\rangle \\
\longrightarrow^{*} & |P|'\langle y := (\lambda x.|M|')\,|N|'\rangle \\
\longrightarrow & |P|'\langle y := |M|'\langle x := |N|'\rangle\rangle
\end{array}
$$

The theorem fails, if we try to reach the second reduct. Here one would need a reduction of the form ($x \notin FV(P)$)

$$P\langle y := M\langle x := N\rangle\rangle \longrightarrow P\langle y := M\rangle\langle x := N\rangle$$

which, as far as we know, has not been considered for any explicit substitution calculus and indeed would be rather strange. □

What is shown by this example is that the obvious candidate for a lambda calculus with an explicit substitution operator and a natural translation from sequent proofs to this lambda calculus gives a closer correspondence between cut-elimination and normalisation, but not all cut-reductions are matched.

We shall encounter a similar phenomenon when trying to extend Theorems 4 and 6 to all connectives. In the $(\wedge, \supset, \forall)$-fragment all $\xrightarrow{c}$-reductions are mapped by $|\_\_|$ onto an identity (see Lemma 2.3). This does not hold for the connectives $\vee$ and $\exists$. In consequence, the correspondence fails. The issues are again best explained with an example.

Example 8 Consider the following sequent proof (where for brevity we have omitted all terms and labels).

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A, B\vdash A\wedge B}\wedge_R \quad \dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A, B\vdash A\wedge B}\wedge_R}{A\vee B, B, A\vdash A\wedge B}\vee_L \quad \dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A\vdash A\wedge A}\wedge_R}{A\wedge B\vdash A\wedge A}\wedge_{L_1}}{A\vee B, B, A\vdash A\wedge A}\,\mathrm{Cut} \tag{2}
$$

To eliminate the cut, we need to apply Rule 8 from Fig. 5 giving

$$
\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A, B\vdash A\wedge B}\wedge_R \quad \dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A\vdash A\wedge A}\wedge_R}{A\wedge B\vdash A\wedge A}\wedge_{L_1}}{A, B\vdash A\wedge A}\,\mathrm{Cut} \quad \dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A, B\vdash A\wedge B}\wedge_R \quad \dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{A\vdash A}}{A\vdash A\wedge A}\wedge_R}{A\wedge B\vdash A\wedge A}\wedge_{L_1}}{A, B\vdash A\wedge A}\,\mathrm{Cut}}{A\vee B, B, A\vdash A\wedge B}\vee_L \tag{3}
$$

where two copies of the cut have been created. The corresponding natural deduction proofs are as follows: forming $|(2)|$ we have the natural deduction proof

$$\dfrac{\dfrac{}{A\vee B\vdash A\vee B}\quad \dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_I \quad \dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_I}{A\vee B,A,B\vdash A\wedge B}\vee_E}{\dfrac{A\vee B,A,B\vdash A}{A\vee B,A,B\vdash A\wedge A}\wedge_I}\wedge_{E_1} \qquad \dfrac{\dfrac{}{A\vee B\vdash A\vee B}\quad \dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_I \quad \dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_I}{A\vee B,A,B\vdash A\wedge B}\vee_E}{\dfrac{A\vee B,A,B\vdash A}{A\vee B,A,B\vdash A\wedge A}\wedge_I}\wedge_{E_1}$$

in which the $\wedge_{E_1}$-rules can be permuted with the $\vee_E$-rules. This gives

$$\dfrac{\dfrac{}{A\vee B\vdash A\vee B}\quad \dfrac{\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_I \quad \dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_I}{A,B\vdash A}\wedge_{E_1} \quad \dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_I \quad \dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_I}{A,B\vdash A}\wedge_{E_1}}{A\vee B,A,B\vdash A}\vee_E}{A\vee B,A,B\vdash A\wedge A}\wedge_I \qquad \dfrac{\dfrac{}{A\vee B\vdash A\vee B}\quad \dfrac{\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_I \quad \dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_I}{A,B\vdash A}\wedge_{E_1} \quad \dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_I \quad \dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_I}{A,B\vdash A}\wedge_{E_1}}{A\vee B,A,B\vdash A}\vee_E}{A\vee B,A,B\vdash A\wedge A}\wedge_I}\wedge_{E_1}$$

Here we are stuck! There is no reduction that would reduce to |(3)|—the natural deduction proof below.

$$\dfrac{\dfrac{}{A\vee B\vdash A\vee B}\quad \dfrac{\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_I}{A,B\vdash A}\wedge_{E_1} \quad \dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_I}{A,B\vdash A}\wedge_{E_1}}{A,B\vdash A\wedge A}\wedge_I \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_I}{A,B\vdash A}\wedge_{E_1} \quad \dfrac{\dfrac{\dfrac{}{A\vdash A}\quad\dfrac{}{B\vdash B}}{A,B\vdash A\wedge B}\wedge_I}{A,B\vdash A}\wedge_{E_1}}{A,B\vdash A\wedge A}\wedge_I}{A\vee B,B,A\vdash A\wedge A}\vee_E$$

What is needed is a reduction rule which permutes $\wedge_I$ with $\vee_E$. On the level of lambda terms the corresponding reduction rule is as follows.

$$\langle \mathtt{case}(P,\lambda x.M,\lambda y.N),\mathtt{case}(P,\lambda z.S,\lambda w.T)\rangle \longrightarrow \mathtt{case}(P,\lambda x.\langle M,S[z/x]\rangle,\lambda y.\langle N,T[w/y]\rangle)$$

□

Again what is shown by this example is that a cut-reduction, which cannot be dispensed with in our cut-elimination procedure, does not match with any of the standard reductions for natural deduction. It requires a reduction which from the term-rewriting point of view is very strange: first it breaks confluence of the lambda-calculus and second it requires a 'test' that checks that two subterms are identical before applying this rule. Nevertheless, from a semantical point of view the rule does make sense: it appears as an equivalence in the work by Altenkirch et al. [1].

5 Conclusion

In this chapter we studied the correspondence question between cut-elimination and normalisation. This was studied previously by Zucker [19]. He showed that in the $(\bigwedge, \supset, \forall)$-fragment of intuitionistic logic reductions sequences in the sequent calculus can be mapped to reduction sequences in natural deduction and vice versa.

Because his cut-elimination procedure is not strongly normalising when $\vee$ and $\exists$ are included, he concluded that the correspondence must fail for these connectives. We obtained a similar result, but using the cut-elimination procedure from Urban [17], which *is* strongly normalising for *all* connectives. We gave a detailed example for why certain cut-reductions do not match with the (standard) reductions of natural deduction. Because we annotated proofs with terms our technical details are much simpler than in Zucker's work. We discovered that there is a rather pleasing interplay between the semantical work by Altenkirch et al. [1] and the correspondence questions. However, details still remain to be investigated.

References

1. Altenkirch, T., Dybjer, P., Hofmann, M., & Scott, P. (2001). Normalization by Evaluation for typed lambda calculus with coproducts. *Proceedings of Logic in Computer Science* (pp. 203–210).
2. Barbanera, F. & Berardi, S. (1994). A symmetric lambda calculus for "classical" program extraction. *Theoretical Aspects of Computer Software, volume 789 of LNCS* (pp. 495–515). Springer.
3. Barendregt, H., & Ghilezan, S. (2000). Theoretical pearls: Lambda terms for natural deduction, sequent calculus and cut elimination. *Journal of Functional Programming*, *10*(1), 121–134.
4. Bloo, R. (1997). Preservation of termination for explicit substitution. *Ph.D. thesis*. Eindhoven University of Technology.
5. Espírito Santo, J. C. (2000). Revisiting the correspondence between cut elimination and normalisation. *Proceedings of ICALP 2000, volume 1853 of LNCS* (pp. 600–611). Springer.
6. Gallier, J. (1993). Constructive logics. Part I: A tutorial on proof systems and typed λ-calculi. *Theoretical Computer Science*, *110*(2), 239–249.
7. Gentzen, G. (1935). Untersuchungen über das logische Schließen I and II. *Mathematische Zeitschrift*, *39*(176–210), 405–431.
8. Girard, J.-Y., Lafont, Y., & Taylor, P. (1989). *Proofs and types, volume 7 of cambridge tracts in theoretical computer science*. Cambridge: Cambridge University Press.
9. Herbelin, H. (1994). A λ-calculus structure isomorphic to sequent calculus structure. *Computer Science Logic, volume 933 of LNCS* (pp. 67–75). Springer.
10. Kreisel, G. (1971). A survey of proof theory II. *Proceedings of the 2nd Scandinavian Logic Symposium, volume 63 of Studies in Logic and the Foundations of Mathematics* pp. 109–170. North-Holland.
11. Laird, J. (2001). A deconstruction of non-deterministic cut elimination. *Proceedings of the 5th International Conference on Typed Lambda Calculi and Applications, Volume 2044 of LNCS* pp. 268–282. Springer.
12. Negri, S., & von Plato, J. (2001). *Structural proof theory*. Cambridge: Cambridge University Press.
13. Pottinger, G. (1977). Normalisation as homomorphic image of cut-elimination. *Annals of Mathematical Logic*, *12*, 323–357.
14. Prawitz, D. (1965). *Natural deduction: A proof-theoretical study*. Stockholm: Almquist and Wiksell.
15. Troelstra, A. S., & Schwichtenberg, H. (2000). *Basic proof theory. Cambridge tracts in theoretical computer science* (2^{nd} ed.). Cambridge: Cambridge University Press.
16. Ungar, A. M. (1992). *Normalisation, cut-elimination and the theory of proofs, volume 28 of CLSI Lecture Notes*. Stanford: Center for the Study of Language and Information.
17. Urban, C. (2000). Classical logic and computation. *Ph.D. Thesis*. Cambridge University.

18. Urban, C., & Bierman, G. M. (2001). Strong normalisation of cut-elimination in classical logic. *Fundamenta Informaticae*, *45*(1–2), 123–155.
19. Zucker, J. (1974). The correspondence between cut-elimination and normalisation. *Annals of Mathematical Logic*, *7*, 1–112.

http://www.springer.com/978-94-007-7547-3

Advances in Natural Deduction
A Celebration of Dag Prawitz's Work
Pereira, L.C.; haeusler, e.; de Paiva, V. (Eds.)
2014, XVI, 279 p. 24 illus., Hardcover
ISBN: 978-94-007-7547-3