Structural Cut Elimination

Frank Pfenning*
Department of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3891

fp@cs.cmu.edu

Abstract

We present new proofs of cut elimination for intuitionistic, classical, and linear sequent calculi. In all cases the proofs proceed by three nested structural inductions, avoiding the explicit use of multi-sets and termination measures on sequent derivations. This makes them amenable to elegant and concise implementations in Elf, a constraint logic programming language based on the LF logical framework.

1 Introduction

Gentzen's sequent calculi [Gen35] for intuitionistic and classical logic have been the central tool in many proof-theoretical investigations and applications of logic in computer science such as logic programming or automated theorem proving. The central property of sequent calculi is *cut elimination* (Gentzen's *Hauptsatz*) which yields consistency of the logic as a corollary. The algorithm for cut elimination may be interpreted computationally, similarly to the way normalization for natural deduction may be viewed as functional computation. For the case of linear logic, this point was made by Girard [Gir87] and later elaborated by Abramsky [Abr93]; see also [BTKP93] and [Gal93]. The study of various cut elimination properties and procedures thus plays an important role in theoretical computer science.

Many proofs of cut elimination have been given in the literature yet, to our knowledge, none of them have been formalized even though this is clearly possible in principle (see Matthews [Mat94] for a pencil-and-paper analysis of cut elimination for the $(\vee, \neg)$ fragment of classical propositional logic in FS_0). They are difficult to mechanize for a number of reasons which in combination are quite intimidating. Most proofs require tedious data structures (such as multi-sets) and use complex termination measures and arithmetic reasoning. They also involve global conditions on occurrences of parameters in sequent derivations. In this paper we present new proofs of cut elimination for intuitionistic, classical, and linear sequent calculi and sketch their implementations in the Elf system [Pfe91] which is based on the LF Logical Framework [HHP93]. Multi-sets are avoided altogether in these proofs, and termination measures are replaced by three nested structural inductions. Parameters are treated as variables bound in derivations, thus naturally capturing occurrence conditions. Critical is the elimination of structural rules, both for the informal proofs and their formalizations, which leads us quite naturally to Kleene's sequent system G_3 [Kle52] for intuitionistic and classical logic and a variant of LU [Gir93] for linear logic. These formulations can easily be seen to be equivalent to more traditional sequent calculi.

The reader interested in structural cut elimination for intuitionistic, classical, or linear logic, but not in its formalization, should be able to follow this paper by ignoring the material regarding implementation. In order to understand and appreciate the representation of the sequent calculus and the proof of cut elimination the reader should have a basic knowledge of the representation methodology of LF and the Elf metalanguage; the interested reader is referred to [HHP93] and [Pfe91].

The principal contributions of this paper are: (1) new formulations of intuitionistic, classical, and linear sequent calculi with proof terms, (2) direct proofs of

*This work was supported by NSF Grant CCR-9303383
This extended abstract to appear at LICS'95

cut elimination for these systems by simple nested structural inductions, (3) extremely concise and elegant implementations of sequent derivations and cut elimination in the Elf meta-language.

The remainder of the paper is organized as follows. In Section 2 we introduce a formulation of the intuitionistic sequent calculus motivated from natural deduction. In Section 3 we give a notation for proof terms that record the structure of the sequent derivation. The representation of sequents in LF based on proof terms is shown in Section 4. The proof of admissibility of cut in the intuitionistic sequent calculus and its implementation are the subject of Section 5. In Section 6 we extend these results to a classical sequent calculus and in Section 7 to a linear sequent calculus. We conclude with an assessment and some remarks about future work in Section 8. Full details for the intuitionistic and classical systems may be found in [Pfe94a]; the linear system is given in [Pfe94b].

2 Intuitionistic Sequent Calculus

In this section we develop a formulation of the sequent calculus for intuitionistic logic by transcribing the process of searching for a natural deduction into an inference system. The proximity to natural deduction then allows a high-level encoding of sequent derivations in LF. The resulting sequent calculus is basically Kleene's system G_3 [Kle52] which he introduced to obtain a simple decidability proof for its propositional fragment. In [Pfe94a] we consider a complete set of logical connectives and quantifiers ($\wedge$, $\supset$, $\vee$, $\neg$, $\top$, $\bot$, $\forall$, and $\exists$). Here we restrict ourselves to $\supset$ and $\exists$ for the sake of brevity. The notions of free and bound variable are defined as usual. We identify formulas that differ only in the names of their bound variables and write $[t/x]A$ for capture-avoiding substitution of t for x in A. We use A, B, and C to range over formulas.

A *sequent* $\Gamma \longrightarrow C$ is a judgment representing the goal of deriving C from Γ. Our formulation eliminates all explicit structural rules (which present well-known difficulties for cut elimination) by incorporating weakening into initial sequents and contraction into each left rule. Exchange remains implicit in the notation Γ, A. We summarize the rules for the cut-free calculus G_3 on our fragment.

$$\frac{}{\Gamma, A \longrightarrow A}\, I \qquad \frac{\Gamma, A \longrightarrow B}{\Gamma \longrightarrow A \supset B}\, \supset R$$

$$\frac{\Gamma, A \supset B \longrightarrow A \qquad \Gamma, A \supset B, B \longrightarrow C}{\Gamma, A \supset B \longrightarrow C}\, \supset L$$

$$\frac{\Gamma \longrightarrow [t/x]A}{\Gamma \longrightarrow \exists x.\ A}\, \exists R \qquad \frac{\Gamma, \exists x.\ A, [a/x]A \longrightarrow C}{\Gamma, \exists x.\ A \longrightarrow C}\, \exists L^a$$

In the $\exists L$ rule, the parameter a must be new, that is, it may not occur in Γ, $\exists x.\ A$, or C. The *principal formula* of an inference is either the formula being introduced on the left or the right, or the formula occurring on the left and the right in an initial sequent. All other formulas are *side formulas* of the last inference. These notions also apply to individual formula occurrences. It is important that the side formulas of an inference are copied to all premises. The rule of cut then has the form

$$\frac{\Gamma \longrightarrow A \qquad \Gamma, A \longrightarrow C}{\Gamma \longrightarrow C}\, \mathrm{Cut}.$$

Instead of taking Cut as a primitive rule, we show that it is admissible, that is, whenever the premises are derivable then the conclusion is also derivable. From this, cut elimination in the sense of Gentzen follows by a simple induction on the structure of a derivation possibly containing Cut (see [Pfe94a]).

Our formulation of the sequent calculus has the elementary properties of weakening (if $\Gamma \longrightarrow C$ then $\Gamma, A \longrightarrow C$), contraction (if $\Gamma, A, A \longrightarrow C$ then $\Gamma, A \longrightarrow C$), and permits substitution for parameters. Note that the rules have been designed so that weakening and contraction do not change the structure of the derivation, only the hypotheses present in each sequent.

At this point we could define the size of a formula A as the number of its connectives and quantifiers, the length of a derivation as the number of inference rules it contains, and then prove the admissibility of cut by three nested inductions over the size of the cut formula and the lengths of the derivations of $\Gamma \longrightarrow A$ and $\Gamma, A \longrightarrow C$. A proof along these general lines, but on a different sequent calculus and with a different rule of cut was given by Dragalin [Dra87]. However, such a proof is not well-suited for implementation. The first difficulty is the encoding of the sequent calculus itself and the notion of multi-set it requires. The second difficulty is that most proof checkers or theorem provers use structural induction more effectively than proofs with termination measures. We will return to both points in the next section.

3 Proof Terms

The sequent rules as given so far do not explicitly indicate the principal formula of an inference. For example, there is only one derivation of $\longrightarrow A \supset (A \supset A)$ rather than two. If we are only interested in derivability then this is tolerable, but not if we need to deal with the *structure* of derivations (as in the case of cut elimination). We therefore endow sequent derivations with proof terms that resolve such ambiguities. Proof terms also play an important role in the study of computational properties of the sequent calculus (see [Abr93, BTKP93, Gal93]), and form a natural intermediate step toward the representation of the rules in LF.

The first step is to label hypotheses. The second is to record a proof term d on the right of the sequent arrow. A sequent then has the form $\Gamma \longrightarrow d : A$ where $\Gamma = h_1{:}A_1, \ldots, h_n{:}A_n$. We assume that all hypothesis labels in a context are distinct. In order to avoid confusion with similar, but subtly different proof term notations in the literature, we systematically introduce precisely one new proof term constructor for each inference rule of the sequent calculus and give each a descriptive name. Rules that introduce parameters or hypotheses bind variables at the level of proof terms—a phenomenon which should be familiar from the Curry-Howard isomorphism. The idea of *higher-order abstract syntax* (here applied to a syntax for proof terms) is to reduce all binding operators to one, namely λ. This makes it immediately syntactically apparent which variables are bound and where. We also indicate the "type" of bound variables: they may bind individuals ($x{:}i$), formulas ($p{:}o$), or hypotheses ($h{:}A$). The rules for the fragment we consider here are given in Figure 1. Propositional parameters p are required in the rules for negation which have been omitted from this extended abstract.

Erasure of the proof terms from a sequent derivation in this calculus yields derivations from the rules given in the previous section. We write $\mathcal{D} :: (J)$ if $\mathcal{D}$ is a derivation of judgment J. Derivations and proof terms are closed under substitutions for parameters, renaming bound variables as necessary to avoid clashes. If $\mathcal{D}$ is a derivation with hypotheses $h_1{:}A$ and $h_2{:}A$, we write $[h_1/h_2]\mathcal{D}$, for the result of erasing hypothesis h_2 on the left-hand side of every sequent occurring in $\mathcal{D}$ and substituting h_1 in every place where h_2 occurs on the right-hand side of a sequent. This may require renaming some locally bound hypotheses to avoid capture of h_1. We write $(\mathcal{D}, h{:}A)$ for the result of adding hypothesis $h{:}A$ to every sequent in $\mathcal{D}$, possibly renaming parameters introduced in $\mathcal{D}$ so as not to conflict with parameters in A. The proofs of the following properties are all immediate structural inductions.

Lemma 1 (Basic Properties of Proof Terms)
The intuitionistic sequent calculus with proof terms satisfies the following properties.

1. *(Weakening) If* $\mathcal{D} :: (\Gamma \longrightarrow d : C)$ *then* $(\mathcal{D}, h{:}A) :: (\Gamma, h{:}A \longrightarrow d : C)$ *where* h *is a new label.*
2. *(Contraction) If* $\mathcal{D} :: (\Gamma, h_1{:}A, h_2{:}A \longrightarrow d : C)$ *then* $[h_1/h_2]\mathcal{D} :: (\Gamma, h_1{:}A \longrightarrow [h_1/h_2]d : C)$.
3. *(Uniqueness) If* $\mathcal{D} :: (\Gamma \longrightarrow d : C)$ *and* $\mathcal{D}' :: (\Gamma \longrightarrow d : C')$ *then* $\mathcal{D} = \mathcal{D}'$ *and* $C = C'$ *(modulo variable renaming).*

4 Sequent Derivations in LF

In this section we briefly summarize the representation of sequent derivations in LF using the idea of higher-order abstract syntax applied to proof terms. Readers interested primarily in the proof of cut elimination may safely skip this section.

For the sake of brevity we show the actual code in Elf [Pfe91], an implementation of LF which permits type declarations with implicit quantifiers. The representation of formulas follows [HHP93]: Object language variables are represented by meta-language variables. This technique is central for this particular encoding; it is not available in systems based on explicit inductive definitions. The type `i` stands for individuals, the type `o` for formulas.

```
i : type.          imp    : o -> o -> o.
o : type.          exists : (i -> o) -> o.
```

Before giving the signature for the sequent calculus we state the adequacy theorem since it is a useful guide for interpreting the declarations. We use $\vdash^{LF}$ for derivability in LF under the signature consisting of the declarations yet to come. Assume we have a derivation

$$\begin{array}{c} \mathcal{D} \\ h_1{:}A_1, \ldots, h_n{:}A_n \longrightarrow d : C \end{array}$$

with free individual parameters among $a_1, \ldots, a_k$ and propositional parameters among $p_1, \ldots, p_m$. Its representation $\ulcorner \mathcal{D} \urcorner$ is a canonical object M such that

$$\begin{array}{l} a_1{:}i, \ldots, a_k{:}i, p_1{:}o, \ldots p_m{:}o, \\ h_1{:}\mathrm{hyp}\ulcorner A_1 \urcorner, \ldots, h_n{:}\mathrm{hyp}\ulcorner A_n \urcorner \vdash^{LF} M : \mathrm{conc}\ulcorner C \urcorner, \end{array}$$

$$\frac{}{\Gamma, h{:}A \longrightarrow \mathrm{axiom}\, h : A}\, I \qquad \frac{\Gamma, h{:}A \longrightarrow d : B}{\Gamma \longrightarrow \mathrm{impr}\, (\lambda h{:}A.\ d) : A \supset B}\, \supset R$$

$$\frac{\Gamma, h{:}A \supset B \longrightarrow d_1 : A \qquad \Gamma, h{:}A \supset B, h_2{:}B \longrightarrow d_2 : C}{\Gamma, h{:}A \supset B \longrightarrow \mathrm{impl}\, d_1\, (\lambda h_2{:}B.\ d_2)\, h : C}\, \supset L$$

$$\frac{\Gamma \longrightarrow d : [t/x]A}{\Gamma \longrightarrow \mathrm{existsr}\, t\, d : \exists x.\ A}\, \exists R \qquad \frac{\Gamma, h{:}\exists x.\ A, h_1{:}[a/x]A \longrightarrow d : C}{\Gamma, h{:}\exists x.\ A \longrightarrow \mathrm{existsl}\, (\lambda a{:}i.\ \lambda h_1{:}[a/x]A.\ d)\, h : C}\, \exists L^a$$

Figure 1: Proof terms for intuitionistic sequent calculus

where hyp and conc are type families indexed by formulas. We call the representation *adequate* if $\ulcorner \cdot \urcorner$ is a bijection between cut-free sequent derivations and such well-typed canonical objects and if it is also *compositional* in sense that $\ulcorner [t/a]\mathcal{D} \urcorner = [\ulcorner t \urcorner / a] \ulcorner \mathcal{D} \urcorner$, $\ulcorner [C/p]\mathcal{D} \urcorner = [\ulcorner C \urcorner / p] \ulcorner \mathcal{D} \urcorner$, and $\ulcorner [h_1/h_2]\mathcal{D} \urcorner = [h_1/h_2] \ulcorner \mathcal{D} \urcorner$.

We obtain the representation by transcribing the proof terms into LF, taking care to distinguish between hypotheses and conclusions via the type families hyp and conc. The signature below represents an adequate encoding of sequents in Elf. Note that `%` begins a comment that extends to the end of the line, that `{x:U}V` is Elf's concrete syntax for $\Pi x{:}U.\ V$, and that `[x:U]M` stands for $\lambda x{:}U.\ M$. Most Π-quantifiers are left implicit and are reconstructed by Elf's front end in proper dependency order and with their most general types.

```
hyp  : o -> type.  % Hypotheses (left)
conc : o -> type.  % Conclusion (right)

axiom : (hyp A -> conc A).

impl  : conc A
         -> (hyp B -> conc C)
         -> (hyp (A imp B) -> conc C).

impr  : (hyp A -> conc B)
         -> conc (A imp B).

existsl : ({a:i} hyp (A a) -> conc C)
           -> (hyp (exists A) -> conc C).

existsr : {T:i} conc (A T)
           -> conc (exists A).
```

The encoding satisfies the representation theorem as outlined above, which means that checking the validity of sequent derivations can be accomplished by type-checking their representations. It circumvents many of the problems that ordinarily arise in representations of the sequent calculus. Multi-sets are avoided, since hypotheses on the left-hand side of the sequent arrow are transported into the LF context. Variable naming conditions are encoded through the functional representation of parametric judgments. The following is proved by inductions on the structure of sequent derivations and canonical forms in LF, using Lemma 1.

Theorem 2 (Adequacy of Representation) *The representation of intuitionistic sequent derivations in LF is adequate.*

5 Admissibility of Cut

The proof of cut elimination uses one principal lemma: the admissibility of cut in the cut-free system. From this, cut elimination follows by a simple structural induction.

Theorem 3 (Admissibility of Cut)

Let $\mathcal{D} :: (\Gamma \longrightarrow d : A)$ and $\mathcal{E} :: (\Gamma, h{:}A \longrightarrow e : C)$ be (cut-free) sequent derivations. Then there exists a proof term f and a (cut-free) sequent derivation $\mathcal{F} :: (\Gamma \longrightarrow f : C)$.

Proof: The proof proceeds by three nested structural inductions on A, d, and e. In other words, we may use the induction hypothesis for (immediate) subformulas of A and arbitrary d and e, or for A, a subterm of d and arbitrary e, and for A, d, and a subterm of e. We distinguish cases for $\mathcal{D}$ and $\mathcal{E}$ (determined by d and e) which can be divided into four categories: (1) Either $\mathcal{D}$ or $\mathcal{E}$ is initial with A as its principal formula, (2) A

is the principal formula of the last inference in both $\mathcal{D}$ and $\mathcal{E}$, (3) A is a side formula of the last inference in $\mathcal{D}$, and (4) A is a side formula of the last inference in $\mathcal{E}$. The proof is constructive and describes an algorithm that computes $\mathcal{F}$ from $\mathcal{D}$ and $\mathcal{E}$. The algorithm is non-deterministic since the cases in the proof are not mutually exclusive. □

This proof is represented as a *relation* between $\ulcorner A \urcorner$`:o`, $\ulcorner \mathcal{D} \urcorner$`:conc A`, $\ulcorner \mathcal{E} \urcorner$`:hyp A -> conc C`, and $\ulcorner \mathcal{F} \urcorner$`:conc C`, which is implemented by a type family

```
ca : {A:o} conc A -> (hyp A -> conc C)
              -> conc C -> type.
```

Each case in the proof gives rise to exactly one declaration for `ca`; no lemmas, auxiliary type families, or theorem proving is needed. We show some excerpts from the signature; in the full language there are 35 cases of comparable complexity comprising 148 lines. The declarations for `ca_axiom_l`, `ca_imp`, and `cal_existsl` implement representative cases of categories (1), (2), and (3), respectively.

```
ca_axiom_l : ca A (axiom H) E (E H).

ca_imp :
  ca (A1 imp A2) (impr D2)
     ([h:hyp (A1 imp A2)] impl (E1 h) (E2 h) h) F
    <- ca (A1 imp A2) (impr D2) E1 E1'
    <- ({h2:hyp A2}
          ca (A1 imp A2) (impr D2)
          ([h:hyp (A1 imp A2)] E2 h h2) (E2' h2))
    <- ca A1 E1' D2 D2'
    <- ca A2 D2' E2' F.

cal_existsl :
  ca A (existsl D1 H) E (existsl D1' H)
    <- ({a:i} {h:hyp (B1 a)}
          ca A (D1 a h) E (D1' a h)).
```

6 Extension to Classical Logic

In sequent calculus, classical logic is handled by allowing multiple conclusions, that is, a sequent has the form $\Gamma \longrightarrow \Delta$, where both Γ and Δ are lists (or multisets) of formulas. For the proof of cut elimination and our representation it is important that Gentzen's structural rules remain implicit: The principal formula of an inference must always be copied to all premises along with all side formulas.

The assignment of proof terms reflects the symmetry between the left- and right-hand sides of a sequent in that we label both negative (left-hand side) and positive (right-hand side) formulas with *variables*. A proof term d then annotates the whole sequent; we write it above the sequent arrow:

$$n_1{:}A_1, \ldots, n_j{:}A_j \xrightarrow{d} p_1{:}C_1, \ldots, p_k{:}C_k.$$

As in the intuitionistic calculus, our proof terms faithfully record the structure of the sequent derivation. We again use λ and the idea of higher-order abstract syntax to delimit scope. Due to space constraints we only show the fragment with $\vee$ and $\neg$ in Figure 2; the development for a full complement of connectives including quantifiers may be found in [Pfe94a].

Substitution, weakening and contraction work as in the intuitionistic case, but now apply to both sides of a sequent. We elide the classical analog of Lemma 1.

The LF representation closely models proof terms and is thus also symmetric with respect to formulas on the left and right: Both appear in the context of the LF typing judgment. That is, a cut-free derivation

$$\begin{array}{c} \mathcal{D} \\ n_1{:}A_1, \ldots, n_j{:}A_j \xrightarrow{d} p_1{:}C_1, \ldots, p_k{:}C_k \end{array}$$

with free individual parameters among $a_1, \ldots, a_m$ is represented by a term $M = \ulcorner \mathcal{D} \urcorner$ such that

$$\begin{array}{l} a_1{:}i, \ldots, a_m{:}i, \\ n_1{:}\mathrm{neg}\ulcorner A_1 \urcorner, \ldots, n_j{:}\mathrm{neg}\ulcorner A_j \urcorner, \\ p_1{:}\mathrm{pos}\ulcorner C_1 \urcorner, \ldots, p_k{:}\mathrm{pos}\ulcorner C_k \urcorner \end{array} \quad \vdash^{LF} M : \#,$$

where neg and pos are type families indexed by formulas and # is a new type, the type of derivations. Below we show the representation of cut-free sequent derivations as an LF signature in the concrete syntax of Elf.

```
# : type.
neg : o -> type.
pos : o -> type.

axiom' : (neg A -> pos A -> #).

orr1'  : (pos A -> #)
           -> (pos (A or B) -> #).

orr2'  : (pos B -> #)
           -> (pos (A or B) -> #).

orl'   : (neg A -> #)
           -> (neg B -> #)
           -> (neg (A or B) -> #).
```

$$\frac{}{\Gamma, n{:}A \xrightarrow{\text{axiom}\,n\,p} p{:}A, \Delta}\, I$$

$$\frac{\Gamma \xrightarrow{d_1} p_1{:}A, p{:}A \vee B, \Delta}{\Gamma \xrightarrow{\text{orr}_1\,(\lambda p_1{:}A.\ d_1)\,p} p{:}A \vee B, \Delta}\, \vee R_1 \qquad \frac{\Gamma, n{:}A \vee B, n_1{:}A \xrightarrow{d_1} \Delta \qquad \Gamma, n{:}A \vee B, n_2{:}B \xrightarrow{d_2} \Delta}{\Gamma, n{:}A \vee B \xrightarrow{\text{orl}\,(\lambda n_1{:}A.\ d_1)\,(\lambda n_2{:}B.\ d_2)\,n} \Delta}\, \vee L$$

$$\frac{\Gamma \xrightarrow{d_2} p_2{:}B, p{:}A \vee B, \Delta}{\Gamma \xrightarrow{\text{orr}_2\,(\lambda p_2{:}B.\ d_2)\,p} p{:}A \vee B, \Delta}\, \vee R_2$$

$$\frac{\Gamma, n{:}A \xrightarrow{d} p{:}\neg A, \Delta}{\Gamma \xrightarrow{\text{notr}\,(\lambda n{:}A.\ d)\,p} p{:}\neg A, \Delta}\, \neg R \qquad \frac{\Gamma, n{:}\neg A \xrightarrow{d} p{:}A, \Delta}{\Gamma, n{:}\neg A \xrightarrow{\text{notr}\,(\lambda p{:}A.\ d)\,n} \Delta}\, \neg L$$

Figure 2: Proof terms for classical sequent calculus

```
notr' : (neg A -> #)
          -> (pos (not A) -> #).

notl' : (pos A -> #)
          -> (neg (not A) -> #).
```

The representation is adequate and compositional; we skip the routine formulation of such a theorem. Admissibility of cut is proved as in the intuitionistic calculus, by three nested structural inductions on the cut formula and the left and right derivations, although there are more cases to consider.

Theorem 4 (Classical Admissibility of Cut)
Let $\mathcal{D} :: (\Gamma \xrightarrow{d} p{:}A, \Delta)$ *and* $\mathcal{E} :: (\Gamma, n{:}A \xrightarrow{e} \Delta)$ *be (cut-free) derivations in the classical sequent calculus* G_3*. Then there is a proof term* f *and a (cut-free) sequent derivation of* $\mathcal{F} :: (\Gamma \xrightarrow{f} \Delta)$*.*

Girard's notion of a *cross-cut* [Gal93], though without unwieldy multiplicities, surfaces naturally in this proof: Since formulas are never discarded they must be eliminated explicitly from both premises of a cut in a cross-cut fashion before the essential cut reduction can take place. The proof is represented by a type family

```
ca' : {A:o} (pos A -> #) -> (neg A -> #)
                -> # -> type.
```

implementing the relation between A, the derivation $\mathcal{D}$, the derivation $\mathcal{E}$ and the resulting derivation $\mathcal{F}$. The implementation of each case is as direct as for the intuitionistic calculus.

7 Extension to Linear Logic

In this section we apply the idea of the encoding for the intuitionistic and classical cases to linear logic. Constructive cut elimination in a sequent formulation is significantly harder here, since structural rules are more difficult to eliminate than in the classical and intuitionistic cases. Girard [Gir87] uses proof nets (instead of a sequent calculus) partly for that reason. Galmiche and Perrier [GP94] give a syntactic analysis of permutabilities of rules and apply it to cut elimination; our own analysis does not go quite as far, but we have a simpler proof of cut elimination. We conjecture that their presentation could be streamlined using our presentation of the linear sequent calculus. Roorda [Roo91] gives a different proof of cut elimination by generalizing the cut rule to multiple occurrences of modal formulas.

The main challenge is to isolate the non-linear reasoning and the associated structural rules. Our solution is close to Andreoli's Σ_2 [And92] and Girard's LU [Gir93] in that we divide a sequent into linear and non-linear zones, and that we have several forms of cut. The structural aspects of non-linear reasoning are treated in the manner of the earlier sections. This leaves a version of dereliction as the only structural rule, and it can be handled directly by the structural induction proving admissibility of cut. A proof along similar lines, but using explicit termination measures instead of structural induction over proof terms has been given by Hodas [Hod94] for $\mathcal{L}$, a fragment of intuitionistic linear logic.

The following fragment is complete for classical propositional linear logic; the extension to include first-order quantifiers and the remaining linear connectives (which are all definable) is straightforward, both in the proof and its implementation in Elf.

$$A ::= P \mid A_1 \otimes A_2 \mid 1 \mid A_1 \,\&\, A_2 \mid \top \mid A^\perp \mid !A \mid ?A$$

We treat $A^\perp$ as primitive and use two-sided sequents to aid in the formalization. We include $?A$ even though it is definable as $(!(A^\perp))^\perp$ due to the special role of the modal operators. A sequent has the form

$$\Psi; \Gamma \longrightarrow \Delta; \Theta$$

which may be interpreted as $!\Psi, \Gamma \longrightarrow \Delta, ?\Theta$ in ordinary linear sequent calculus. Thus the outer zones in the sequents represent non-linear hypotheses and conclusions, the inner zones must be treated linearly. The calculus is defined by the rules in Figure 3.

There are three rules of cut, which we show to be admissible, rather than taking them as primitive. These rules take advantage of the additional structure provided by the multi-zonal sequent, but they are pure in the sense that do not refer directly to the exponentials of linear logic.

$$\frac{\Psi; \Gamma_1 \longrightarrow A, \Delta_1; \Theta \qquad \Psi; \Gamma_2, A \longrightarrow \Delta_2; \Theta}{\Psi; \Gamma_1, \Gamma_2 \longrightarrow \Delta_1, \Delta_2; \Theta}\,\text{Cut}$$

$$\frac{\Psi; \cdot \longrightarrow A; \Theta \qquad (\Psi, A); \Gamma \longrightarrow \Delta; \Theta}{\Psi; \Gamma \longrightarrow \Delta; \Theta}\,\text{Cut!}$$

$$\frac{\Psi; \Gamma \longrightarrow \Delta; (A, \Theta) \qquad \Psi; A \longrightarrow \cdot; \Theta}{\Psi; \Gamma \longrightarrow \Delta; \Theta}\,\text{Cut?}$$

This system satisfies weakening and contraction in the non-linear zones: We can adjoin a formula to the non-linear zones of each sequent in a derivation to achieve weakening, or substitute the use of one formula for another to achieve contraction. In either case the structure of the derivation does not change.

We could now prove admissibility of these rules simultaneously by three nested inductions on the complexity of the cut formula, the length of the derivation of the left premise and the length of the derivation of the right premise. However, for the purpose of implementation we would like the proof to be structural. As before, we introduce proof terms so that the required weakening for formulas in the non-linear zones does not destroy the structural induction. However, there are two new phenomena: proof terms now come from a *linear λ-calculus* and they no longer uniquely determine the sequent derivation, since the side formulas of multiplicative rules may be split differently without affecting the proof term. For our purposes, proof terms for the Lolli fragment of linear logic [HM94] are sufficient, which is important since it satisfies a stronger normal form theorem than the full calculus. In this fragment, we have linear (`-o`) and intuitionistic (`->`) implication, additive conjunction (`&`), and top (`T`) and corresponding proof constructors. In Figure 4 we show the declarations of the constants that could be used to construct the linear λ-terms in the representation. Alternatively, one can think of this as a specification in Lolli where each clause has been labelled.

Lolli is first-order and has no notion of proof terms and is thus not adequate to obtain an implementation of derivations in the sequent calculus, only of derivability. LF, on the other hand, has no notion of linearity. Thus we approximate the proof terms as sketched above in LF by interpreting `-o` as `->` and eliminating `T` (top) and `&` by currying, basically ignoring linearity restrictions, but properly modelling dependencies and variable scope. This representation is not yet adequate, but we can recover adequacy by using a higher-level judgment `lin` on proof terms, checking that they are linear. This requires two auxiliary judgments checking that embedded functions are linear in their argument where required. We only show the declarations—the implementation is relatively straightforward.

```
linp : (pos A -> #) -> type.
linn : (neg A -> #) -> type.
lin  : # -> type.
```

The resulting representation of derivations is a compositional bijection between between LF objects of type `lin` in an appropriate context and linear sequent derivations in LV (a precise formulation of this theorem may be found in [Pfe94b]). We can prove admissibility of cut by nested structural inductions as before.

Theorem 5 (Admissibility of Cut in LV) *The three rules of cut for the linear sequent calculus are admissible.*

Proof: Let A be the cut formula and d and e the proof terms of the derivations of the premisses of the cut rules. We proceed by three nested structural inductions on A, d and e, simultaneously for Cut, Cut!,

and Cut?. Appeals to Cut! and Cut? are considered greater than Cut, if the cut formula A is the same. To rephrase: We may appeal to the induction hypothesis on (1) a smaller cut formula, (2) the same cut formula, but pass from Cut! or Cut? to Cut, (3) the same cut formula and rule, but smaller proof term d, or (4) the same cut formula, rule, proof terms d, but smaller proof term e. □

The faithful and concise implementation of this proof in the spirit of the earlier encodings would require a linear logical framework, which is the subject of current research. But we can implement the transformation on proof terms as they are represented in LF. Note that the reasoning about linearity restrictions in the structural proof is not implemented here. Nonetheless, it can be executed by applying it to proof terms to obtain proof terms for the result of the cut. By meta-level reasoning (*i.e.*, the informal proof and properties of LF) we know that this will always map linear derivations to linear derivations. We only show the declarations and three typical cases in Figure 5.

8 Conclusion

We have presented new structural proofs of cut elimination for intuitionistic, classical, and linear sequent calculi. We have further shown how sequent derivations in all three systems can be implemented in Elf. For the intuitionistic and classical cases, the proof of cut elimination can also be implemented in Elf, although the fact that this implementation models the informal argument is still partly an informal property, just like the adequacy of the LF encoding of derivations. The implementations can be executed to transform sequent derivations with cut to cut-free derivations. The proof representation is extremely concise and much shorter than an informal proof of the same argument. We have written a program to generate LaTeX source for an informal version of each case in the proof of admissibility. These "informalized" versions are given in full detail in [Pfe94a].

In the case of linear logic the implementation of cut elimination captures less of the informal reasoning by ignoring linearity constraints, but is nonetheless operationally adequate in the sense that, given valid linear sequent derivations, it will generate valid linear sequent derivations. The linearity check has also been implemented in Elf, although it is somewhat tedious. The combined sources for the implementations of admissibility of cut and cut elimination for intuitionistic and classical logic with a full complement of quantifiers and connectives comprise 739 lines of Elf code and require about 2 seconds to type-check on a Dec Alpha. Cut elimination for linear logic adds 675 lines, which includes the algorithm comprising 73 cases. Once we arrived at the basic representation idea, the full implementation was carried out, revised, and cleaned up in about 2 days each for intuitionistic, classical, and linear sequent calculi.

Once the structural proof of admissibility has been found and implemented, it is natural to ask if it can also be encoded in stronger frameworks such as Coq [DFH+93] so that structural inductions are made explicit and the proof is fully formally verified. There are several aspects of our proof which make this difficult. The first is the essential use of higher-order abstract syntax, which is not available in a similarly straightforward fashion in other candidate environments. The second difficulty arises from the nondeterministic nature of the cut elimination algorithm contained in the proof. Making it deterministic in the form of a primitive recursion (which would be required for a functional framework) would lead to an explosion in the number of cases. It appears the only way to avoid at least some of this combinatorial explosion is to introduce termination measures after all, which requires a new sequence of lemmas regarding sizes of formulas and derivations. We conclude that a similarly elegant representation of cut elimination in other systems is a non-trivial challenge which, we hope, others will take up.

In future work we plan to verify mechanically that the given signatures indeed implement proofs. The prototype implementation of the schema-checker sketched in [Roh94] currently accepts them, but the (meta-meta-)theoretical analysis of the schema-checker itself is not yet complete. In other future work we plan to reexamine the connection between normalization and cut elimination in the same framework. Another direction is to study cut elimination in a formulation as a higher-order rewrite system along the lines of Nipkow [Nip91], but using dependent types. We first note that our system of rules is terminating (note that we cannot permute adjacent cuts). Assuming the completeness of a critical pair criterion for the dependently typed calculus, the system is confluent modulo Kleene's permutations of adjacent inference rules in the cut-free system. This means that our cut conversions do not identify intuitively unrelated sequent derivations, which has been a problem in other systems as noted by Lafont (see [Gal93]).

Finally, the representation of the linear sequent calculus and its cut elimination algorithm could be made

even more concise by using a linear logical framework as sketched in Section 7. In joint work with I. Cervesato, we are currently designing such a linear refinement of LF following ideas of Miller, Plotkin and Pym [MPP92].

Acknowledgments. I would like to thank Iliano Cervesato, Jean Gallier, Dale Miller, Richard Statman and Roberto Virga for discussions regarding the topic of this paper and David Basin, Seán Matthews, Ekkehard Rohwedder and Anne Troelstra for comments on a draft.

References

[Abr93] Samson Abramsky. Computational interpretations of linear logic. *Theoretical Computer Science*, 111:3–57, 1993.

[And92] Jean-Marc Andreoli. Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation*, 2(3):197–347, 1992.

[BTKP93] Val Breazu-Tannen, Delia Kesner, and Laurence Puel. A typed pattern calculus. In Moshe Y. Vardi, editor, *Proceedings of the Eight Annual IEEE Symposium on Logic in Computer Science*, pages 262–274, Montreal, Canada, June 1993.

[DFH+93] Gilles Dowek, Amy Felty, Hugo Herbelin, Gérard Huet, Chet Murthy, Catherine Parent, Christine Paulin-Mohring, and Benjamin Werner. The Coq proof assistant user's guide. Rapport Techniques 154, INRIA, Rocquencourt, France, 1993. Version 5.8.

[Dra87] A. G. Dragalin. *Mathematical Intuitionism: Introduction to Proof Theory*, volume 67 of *Translations of Mathematical Monographs*. American Mathematical Society, Providence, Rhode Island, 1987.

[Gal93] Jean Gallier. Constructive logics Part I: A tutorial on proof systems and typed λ-calculi. *Theoretical Computer Science*, 110:249–339, 1993.

[Gen35] Gerhard Gentzen. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1935. English translation in M. E. Szabo, editor, *The Collected Papers of Gerhard Gentzen*, pages 68–131, North-Holland, 1969.

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[Gir93] Jean-Yves Girard. On the unity of logic. *Annals of Pure and Applied Logic*, 59:201–217, 1993.

[GP94] Didier Galmiche and Guy Perrier. On proof normalization in linear logic. *Theoretical Computer Science*. To appear. Available as Technical Report CRIN 94-R-113, Nancy, France.

[HHP93] Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. *Journal of the Association for Computing Machinery*, 40(1):143–184, January 1993.

[HM94] Joshua Hodas and Dale Miller. Logic programming in a fragment of intuitionistic linear logic. *Information and Computation*, 110(2):327–365, 1994. A preliminary version appeared in the Proceedings of the Sixth Annual IEEE Symposium on Logic in Computer Science, pages 32–42, Amsterdam, The Netherlands, July 1991.

[Hod94] Joshua S. Hodas. *Logic Programming in Intuitionistic Linear Logic: Theory, Design, and Implementation*. PhD thesis, University of Pennsylvania, Department of Computer and Information Science, 1994.

[Kle52] Stephen Cole Kleene. *Introduction to Metamathematics*. North-Holland, 1952.

[Mat94] Seán Matthews. A theory and its metatheory in FS_0. In Dov Gabbay and Franz Guenthner, editors, *What is a Logical System?* Oxford University Press, 1994. To appear.

[MPP92] Dale Miller, Gordon Plotkin, and David Pym. A relevant analysis of natural deduction. Talk given at the Workshop on Logical Frameworks, Båstad, Sweden, May 1992.

[Nip91] Tobias Nipkow. Higher-order critical pairs. In G. Kahn, editor, *Sixth Annual IEEE Symposium on Logic in Computer Science*, pages 342–349, Amsterdam, The Netherlands, July 1991.

[Pfe91] Frank Pfenning. Logic programming in the LF logical framework. In Gérard Huet and Gordon Plotkin, editors, *Logical Frameworks*, pages 149–181. Cambridge University Press, 1991.

[Pfe94a] Frank Pfenning. A structural proof of cut elimination and its representation in a logical framework. Technical Report CMU-CS-94-218, Department of Computer Science, Carnegie Mellon University, November 1994.

[Pfe94b] Frank Pfenning. Structural cut elimination in linear logic. Technical Report CMU-CS-94-222, Department of Computer Science, Carnegie Mellon University, December 1994.

[Roh94] Ekkehard Rohwedder. Verifying the metatheory of deductive systems. Thesis Proposal, February 1994.

[Roo91] Dirk Roorda. *Resource Logics: Proof-Theoretical Investigations*. PhD thesis, University of Amsterdam, September 1991.

$$\frac{}{\Psi; A \longrightarrow A; \Theta} I$$

$$\frac{\Psi; \Gamma_1 \longrightarrow A, \Delta_1; \Theta \qquad \Psi; \Gamma_2 \longrightarrow B, \Delta_2; \Theta}{\Psi; \Gamma_1, \Gamma_2 \longrightarrow A \otimes B, \Delta_1, \Delta_2; \Theta} \otimes R \qquad \frac{\Psi; \Gamma, A, B \longrightarrow \Delta; \Theta}{\Psi; \Gamma, A \otimes B \longrightarrow \Delta; \Theta} \otimes L$$

$$\frac{}{\Psi; \cdot \longrightarrow 1; \Theta} 1R \qquad \frac{\Psi; \Gamma \longrightarrow \Delta; \Theta}{\Psi; \Gamma, 1 \longrightarrow \Delta; \Theta} 1L$$

$$\frac{\Psi; \Gamma \longrightarrow A, \Delta; \Theta \qquad \Psi, \Gamma \longrightarrow B, \Delta; \Theta}{\Psi; \Gamma \longrightarrow A \,\&\, B, \Delta; \Theta} \,\&\, R \qquad \frac{\Psi; \Gamma, A \longrightarrow \Delta; \Theta}{\Psi; \Gamma, A \,\&\, B \longrightarrow \Delta; \Theta} \,\&\, L_1 \qquad \frac{\Psi; \Gamma, B \longrightarrow \Delta; \Theta}{\Psi; \Gamma, A \,\&\, B \longrightarrow \Delta; \Theta} \,\&\, L_2$$

$$\frac{}{\Psi; \Gamma \longrightarrow \top, \Delta; \Theta} \top R \qquad \text{No } \top L \text{ rule}$$

$$\frac{\Psi; \Gamma, A \longrightarrow \Delta; \Theta}{\Psi; \Gamma \longrightarrow A^{\perp}, \Delta; \Theta} \neg R \qquad \frac{\Psi; \Gamma \longrightarrow A, \Delta; \Theta}{\Psi; \Gamma, A^{\perp} \longrightarrow \Delta; \Theta} \neg L$$

$$\frac{\Psi; \cdot \longrightarrow A; \Theta}{\Psi; \cdot \longrightarrow !A; \Theta} !R \qquad \frac{(\Psi, A); \Gamma \longrightarrow \Delta; \Theta}{\Psi; (\Gamma, !A) \longrightarrow \Delta; \Theta} !L$$

$$\frac{\Psi; \Gamma \longrightarrow \Delta; (A, \Theta)}{\Psi; \Gamma \longrightarrow (?A, \Delta); \Theta} ?R \qquad \frac{\Psi; A \longrightarrow \cdot; \Theta}{\Psi; ?A \longrightarrow \cdot; \Theta} ?L$$

$$\frac{(\Psi, A); (\Gamma, A) \longrightarrow \Delta; \Theta}{(\Psi, A); \Gamma \longrightarrow \Delta; \Theta} !D \qquad \frac{\Psi; \Gamma \longrightarrow (A, \Delta); (A, \Theta)}{\Psi; \Gamma \longrightarrow \Delta; (A, \Theta)} ?D$$

Figure 3: Rules for LV

```
# : type.         % Token for Derivations
neg!: o -> type.  % Modal Hypotheses (far left)
neg : o -> type.  % Hypotheses (left)
pos : o -> type.  % Conclusions (right)
pos?: o -> type.  % Modal Conclusions (far right)

axiom : (neg A -o pos A -o #).

timesr : (pos A -o #)                    timesl : (neg A -o neg B -o #)
          -o (pos B -o #)                          -o (neg (A times B) -o #).
          -o (pos (A times B) -o #).

oner : (pos one -o #).                   onel : # -o (neg one -o #).

                                         andl1 : (neg A -o #)
andr : ((pos A -o #) & (pos B -o #))              -o (neg (A and B) -o #).
        -o (pos (A and B) -o #).
                                         andl2 : (neg B -o #)
                                                  -o (neg (A and B) -o #).

topr : T -o (pos (top) -o #).            % no topl

perpr : (neg A -o #)                     perpl : (pos A -o #)
         -o (pos (perp A) -o #).                  -o (neg (perp A) -o #).

!r : (pos A -o #)                        !l : (neg! A -> #)
      -> (pos (! A) -o #).                     -o (neg (! A) -o #).

?r : (pos? A -> #)                       ?l : (neg A -o #)
      -o (pos (? A) -o #).                     -> (neg (? A) -o #).

!d : (neg A -o #)                        ?d : (pos A -o #)
      -o (neg! A -> #).                        -o (pos? A -> #).
```

Figure 4: Implementation of LV

```
ad  : {A:o} (pos A -> #) -> (neg A -> #) -> # -> type.
ad! : {A:o} (pos A -> #) -> (neg! A -> #) -> # -> type.
ad? : {A:o} (pos? A -> #) -> (neg A -> #) -> # -> type.

ad_times :
   ad (A times B) ([p] timesr D1 D2 p) ([n] timesl E1 n) F
    <- ({n2:neg B} ad A D1 ([n1:neg A] E1 n1 n2) (E1' n2))
    <- ad B D2 E1' F.

ad!_d :
   ad! A D! ([n!] !d (E1 n!) n!) F
    <- ({n1:neg A} ad! A D! ([n!] E1 n! n1) (E1' n1))
    <- ad A D! E1' F.

adl_andr :
   ad A ([p] andr (D1 p) (D2 p) P) E (andr D1' D2' P)
    <- ({p1:pos B1} ad A ([p] D1 p p1) E (D1' p1))
    <- ({p2:pos B2} ad A ([p] D2 p p2) E (D2' p2)).
```

Figure 5: Admissibility of cut in LV (three cases)