On the specification of sequent systems

Elaine Pimentel[1] and Dale Miller[2⋆]

[1] Departamento de Matemática,
Universidade Federal de Minas Gerais, Belo Horizonte, M.G. Brasil
[2] INRIA-Futurs & Laboratoire d'Informatique (LIX)
École Polytechnique France

Abstract. Recently, linear Logic has been used to specify sequent calculus proof systems in such a way that the proof search in linear logic can yield proof search in the specified logic. Furthermore, the meta-theory of linear logic can be used to draw conclusions about the specified sequent calculus. For example, derivability of one proof system from another can be decided by a simple procedure that is implemented via bounded logic programming-style search. Also, simple and decidable conditions on the linear logic presentation of inference rules, called *homogeneous* and *coherence*, can be used to infer that the initial rules can be restricted to atoms and that cuts can be eliminated. In the present paper we introduce *Llinda*, a logical framework based on linear logic augmented with inference rules for definition (fixed points) and induction. In this way, the above properties can be proved entirely inside the framework. To further illustrate the power of *Llinda*, we extend the definition of coherence and provide a new, semi-automated proof of cut-elimination for Girard's Logic of Unicity (LU).

1 Introduction

Logics and type systems have been exploited in recent years as frameworks for the specification of deduction in a number of logics. Such *meta-logics* or *logical frameworks* have been mostly based on intuitionistic logic (see, for example, [FM88,NM88,Har93]) or dependent types (see [Pfn89]) in which quantification at (non-predicate) higher-order types is available. These computer systems have been used as meta-languages to automate various aspects of different logics.

Features of a meta-logic are often directly inherited by any object-logic. This inheritance can be, at times, a great asset: for example, the meta-logic treatment of binding and substitution can be exploited directly in specifying the object-logic. On the other hand, features of the meta-logic can limit the kinds of object-logics that can be directly and naturally encoded. For example, the structural rules of an intuitionistic meta-logic (weakening and contraction) are also inherited and make it difficult to have natural encodings of logics for which these structural rules are not intended. Also, intuitionistic logic does not have

⋆ This work has been supported in part by the ACI grants Geocal and Rossignol and the INRIA "Equipes Associées" Slimmer.

an involutive negation and this makes it difficult to address directly dualities in object-logic proof systems. This lack of dualities is particularly unfortunate when specifying sequent calculus [Gen69] since they play a central role in the theory of such proof systems.

Pfenning in [Pfn95,Pfn00] used the logical framework LF to give new proofs of cut elimination for intuitionistic and classical sequent calculi. His approach is elegant since many technical details of the cut-elimination proof were aborbed by the LF. That approach, however, is based on an intuitionistic meta-logic and is not so suitable for handling the dualities of the sequent calculus.

In [Mil96,MP04,MP02], classical linear logic was used as a meta-logic in order to specify and reason about a variety of proof systems. Since the encodings of such logical systems are natural and direct, the meta-theory of linear logic can be used to draw conclusions about the object-level proof systems. More specifically, in [MP02], the authors present a decision procedure for determining if one encoded proof system is derivable from another. In the same paper, necessary conditions were presented (together with a decision procedure) for assuring that an encoded proof system satisfies cut-elimination. This last result used linear logic's dualities to formalize the fact that if the left and right introduction rules are suitable duals of each other then non-atomic cuts can be eliminated.

In the present paper, we go a step further and introduce *Llinda*, a logical framework based on linear logic augmented with inference rules for definition (fixed points) and induction. In this stronger logic, such properties on an object-logic as the elimination of non-atomic cuts can be proved entirely inside the logical framework. In particular, much of the meta-reasoning that appears in [MP02] can be internalized in *Llinda*. We also use *Llinda* to give sufficient and decidable conditions that guarantee the completeness of the atomic initial rule. Many consider, as Girard [Gir99], that such a property is a crucial condition when designing a "good sequent system". To further illustrate the power of *Llinda* as a framework for specifying and reasoning about sequent systems, we extend the definition of coherence [MP02] and provide a new, semi-automated proof of cut-elimination for LU, Girard's Logic of Unicity [Gir93].

The rest of the paper is organized as follows. Section 2 introduces the notion of flat linear logic and Section 3 extends linear logic with definitions and induction. Section 4 presents a method for encoding logical rules and Section 5 represents introduction rules as definitions. Section 6 highlights the role of bipolar formulas in the specification of sequent systems. Section 7 presents a necessary condition for characterizing systems having the cut-elimination property while in Section 8 a necessary condition is given that guarantees that initial rules can be restricted to atomic formulas. Finally, Section 9 presents a semi-automated proof of cut-elimination for LU.

2 Flat Linear Logic

The connectives of linear logic [Gir87] can be classified as *synchronous* and *asynchronous* [And92]: the asynchronous connectives have right-introduction rules

that are invertible while the right-introduction rules of *synchronous* connective are not generally invertible and they usually require "synchronization" between the introduced formula and its context within a sequent. The de Morgan dual of a connective in one class yields a connective in the other class.

Although full linear logic is important in this work, we need to consider certain formulas of rather restricted nesting of synchronous and asynchronous connectives. These restricted formulas will carry the adjective "flat".

Definition 1. *A* flat goal *is a linear logic formula that contains only occurrences of the asynchronous connectives (namely* $\parr, \&, \bot, \top, \forall$*) together with the modal* ? *which can only have atomic scope. A* flat clause *is a linear logic formula of the form:*

$$\forall \bar{y}(G_1 \hookrightarrow \cdots \hookrightarrow G_m \hookrightarrow A_1 \parr \cdots \parr A_n), \quad (m, n \geq 0)$$

where $G_1, \ldots, G_m$ *are flat goals,* $A_1, \ldots, A_n$ *are atomic formulas and occurrences of* $\hookrightarrow$ *represent either* $\multimap$ *or* $\Rightarrow$*. The formula* $A_1 \parr \cdots \parr A_n$ *is the* head *of such a clause, while for each* $i = 1, \ldots, m$*, the formula* G_i *is a* body *of this clause. If* $n = 0$*, then we write the head simply as* $\bot$ *and say that the head is* empty*.*

A flat clause is logically equivalent to a formula in *uncurried* form, namely, a formula of the form

$$\forall \bar{y}(B \multimap A_1 \parr \cdots \parr A_n)$$

where $n \geq 0$, $\bar{y}$ is the list of variables free in the head $A_1 \parr \cdots \parr A_n$, all free variables of B are also free in the head, and B may have outermost occurrences of the synchronous connectives: 1, $\oplus$, $\otimes$, $\exists$ and !. We will call B an *uncurried flat body.*

A formula that is either a flat goal or a uncurried flat body is an example of a *bipolar* formula, namely, a formula in which no synchronous connective is in the scope of an asynchronous connective.

As in Church's Simple Theory of Types [Chu40], types for both terms and formulas are built using a simply typed λ-calculus. Variables are simply typed that do not contain the type o, which is reserved for the type of formulas. We will call types which do not contain the type o *object types*, and variables and constants of object types are named object variables and object constant, respectively. Otherwise types will be referred as *meta-level types* and formulas will be called meta-level formulas. We assume the usual rules of α, β, and η-conversion and we identify terms and formulas up to α-conversion. A term is λ-normal if it contains no β and no η redexes. All terms are λ-convertible to a term in λ-normal form, and such a term is unique up to α-conversion. The substitution notation $B[t/x]$ denotes the λ-normal form of the β-redex $(\lambda x.B)t$.

3 *Llinda*: Linear logic with definition and induction

Following the lines described by McDowell and Miller [MM00] and Tiu [Tiu04] on the proof theoretic notion of definitions, we will extend linear logic by allowing the definition of atomic formulas.

Definition 2. *A definition $\mathcal{D}$ is a finite set of* definition clauses*, which are expressions of the form $\forall\bar{x}[p\bar{x} \stackrel{\triangle}{=} B\bar{x}]$, where p is a predicate constant. The formula $B\bar{x}$ is the* body *and the atomic formula $p\bar{x}$ is the* head *of that clause. A predicate may occur at most once in the heads of the clauses of a definition.*

The symbol $\stackrel{\triangle}{=}$ is not a logical connective: it simply marks a definition clause.

Linear logic augmented with such *definitions* is not consistent if these definitions are not restricted. For instance, if negative occurrences of the exponential ! are allowed in the body of definitions, inconsistencies can be easily constructed. In order to avoid such inconsistencies, we introduce the notion of level of a formula to define a proper stratification on definitions, as done in [MM00,Tiu04]. To each predicate p we associate a natural number $lvl(p)$, the level of p. The notion of level is then extended to formulas.

Definition 3. *Given a formula B, its level $lvl(B)$ is defined as follows:*

1. $lvl(p\bar{t}) = lvl(p)$
2. $lvl(\bot) = lvl(\top) = lvl(1) = lvl(0) = 0$
3. $lvl(!\,A) = lvl(?\,A) = lvl(A)$
4. $lvl(B \oplus C) = lvl(B \parr C) = lvl(B \,\&\, C) = lvl(B \otimes C) = max(lvl(B); lvl(C))$
5. $lvl(\forall x.A) = lvl(\exists x.A) = lvl(A)$
6. $lvl(A_1 \multimap A_2) = max(lvl(A_1) + 1; lvl(A_2))$.

Definition 4. *A definition clause $\forall\bar{x}.[p\bar{x} \stackrel{\triangle}{=} B]$ is* stratified *if $lvl(B) \leq lvl(p)$. A definition is* stratified *if all its definition clauses are stratified. An occurrence of a formula A in a formula C is* strictly positive *if that particular occurrence of A is not to the left of any implication in C. In this way, the stratification of definitions implies that for every definition clause all occurrences of the head predicate in the body are strictly positive.*

Observe that stratification excludes the possibility of circular calling through implications (negations). Since all occurrences of p in B are *positive*,the existence of fixed points is always guaranteed. Thus the provability of pt means that t is in a solution of the corresponding fixed point equation.

Note also that a flat clause that is written in its uncurried form can be seen as a definition clause since uncurried bodies are uncurried flat goals (and hence do not contain implications).

Definition 5. *A definition clause $\forall\bar{x}.[p\bar{x} \stackrel{\triangle}{=} B]$ is* flat *if B is an uncurried flat body. A definition is* flat *if all its definition clauses are flat.*

Given a definition clause $\forall\bar{x}[p\bar{x} \stackrel{\triangle}{=} B\bar{x}]$, the left and right rules for atoms are

$$\frac{B\bar{t}, \Delta \longrightarrow \Gamma}{p\bar{t}, \Delta \longrightarrow \Gamma}\ def\mathcal{L} \qquad \frac{\Delta \longrightarrow B\bar{t}, \Gamma}{\Delta \longrightarrow p\bar{t}, \Gamma}\ def\mathcal{R}.$$

The rules above show that an atom can be substituted by its definition during a proof. This means that a defined atom can be seen as a generalized connective, whose behavior is determined by its defining clause.

Since a predicate may occur at most once in the heads of definitions, explicit equality must appear as part of the syntax. The rules for the equality predicate makes use of (the standard notion of) substitutions. The left and right introduction rules for equality are:

$$\frac{\{\Gamma\theta \longrightarrow \Delta\theta \mid s\theta =_{\beta,\eta} t\theta,\ \theta \in CSU(s,t)\}}{(s=t), \Gamma \longrightarrow \Delta}\ eq\mathcal{L} \qquad \frac{}{\longrightarrow t=t}\ eq\mathcal{R}.$$

The set $CSU(s,t)$ is a *complete set of unifiers* for s and t. In general, $CSU(s,t)$ can be empty (for non-unifiability), finite, or infinite. Thus the set of sequents as the premise in the $eq\mathcal{L}$ rule should be understood to mean that each sequent in the set is a premise of the rule. Notice that in the $eq\mathcal{L}$ rule, the free variables of the conclusion can be instantiated in the premises. In the examples in this paper, the set $CSU(s,t)$ can be taken as being either empty or a singleton, containing the most general unifier of s and t.

As observed before, a definition $\forall x.px \triangleq Bx$ can be seen as a fixed point equation, but that fixed point is not necessarily the least or the greatest one. We now add extra rules for capturing the least fixed point via *induction*.

Let $\forall\bar{x}[p\bar{x} \triangleq B\bar{x}]$ be a stratified definitional clause and let S be a closed term of the same type as p. The left introduction rule for an atom with predicate p can be strengthed to be

$$\frac{(B\bar{x})[S/p] \longrightarrow S\bar{x} \quad \Delta, S\bar{t} \longrightarrow \Gamma}{\Delta, p\bar{t} \longrightarrow \Gamma}\ ind\mathcal{L}.$$

The formula S is an invariant of the induction and it is called the *inductive predicate*. The variables $\bar{x}$ are new eigenvariables. The expression $(B\bar{x})[S/p]$ denotes the result of replacing the predicate p in $B\bar{x}$ with S (and λ-normalizing).

Definition 6. *Llinda is linear logic with stratified definition and induction.*[1]

A sequent in *Llinda* will be represented as $\mathcal{D} \parallel \Delta \longrightarrow \Gamma$, meaning the linear sequent with the set of definitions $\mathcal{D}$. If the definition is empty or when it is clear from the context, we will write the sequent above as the usual linear sequent $\Delta \longrightarrow \Gamma$.

We introduce the natural numbers via the type nt, the constants $z : nt$ for zero and $s : nt \to nt$ for successor function and the inductive predicate $nat : nt \to o$, with the following definition clause:

$$nat\ x \triangleq [x = z] \oplus \exists y.[x = sy \otimes nat\ y].$$

Proposition 1. *The following rules can be derived in Llinda:*

$$\frac{\longrightarrow B\,z \quad B\,i \longrightarrow B\,(s\,i) \quad B\,I, \Delta \longrightarrow \Gamma}{nat\ I, \Delta \longrightarrow \Gamma}\ nat\mathcal{L}$$

[1] The word "*linda*", in Portuguese, means "extremely beautiful."

$$\frac{!\,\Delta \longrightarrow B\,z, ?\,\Gamma \quad !\,\Delta, B\,j \longrightarrow B\,(s\,j), ?\,\Gamma \quad B\,I, !\,\Delta, \Delta' \longrightarrow \Gamma', ?\,\Gamma}{nat\,I, !\,\Delta, \Delta' \longrightarrow \Gamma', ?\,\Gamma}$$

$$\frac{\longrightarrow B \quad B, \Delta \longrightarrow \Gamma}{nat\,I, \Delta \longrightarrow \Gamma} \qquad \frac{\Delta \longrightarrow \Gamma}{nat\,I, \Delta \longrightarrow \Gamma} \qquad \forall n[nat\,n \equiv !\,nat\,n]$$

For an example of specifying an object-logic, consider intuitionistic logic over the following logical connectives: $\cap$, $\cup$, f_i, and t_i for conjunction, disjunction, false, and true; $\supset$ for implication, and $\forall_i$ and $\exists_i$ for universal and existential quantification. Now introduce the type *bool* of intuitionistic formulas and the inductive predicate $form_i(\cdot) : bool \to o$ with the following defined clause:

$$\begin{array}{lll} form_i(x) \triangleq & [x = t_i] & \oplus \\ & [x = f_i] & \oplus \\ & atomic(x) & \oplus \\ & \exists y, w.[(x = y \cap w) \otimes form_i(y) \otimes form_i(w)] & \oplus \\ & \exists y, w.[(x = y \cup w) \otimes form_i(y) \otimes form_i(w)] & \oplus \\ & \exists y, w.[(x = y \supset w) \otimes form_i(y) \otimes form_i(w)] & \oplus \\ & \exists X.[(x = \forall_i u.X\,u) \otimes (\forall u.form_i(X\ u))] & \oplus \\ & \exists X.[(x = \exists_i u.X\,u) \otimes (\forall u.form_i(X\ u))] & \end{array}$$

The predicate *atomic* is given elsewhere as a definition. The *ind*$\mathcal{L}$ rule applied to this definition yields an induction principle for object-level formulas. Following the same arguments used above for natural numbers, it is possible to derive the following, more intuitive rule for structural induction.

Proposition 2. *The following rule can be derived in Llinda*

$$\frac{\begin{array}{c} \longrightarrow B\,t_i \qquad \longrightarrow B\,f_i \qquad atomic(x) \longrightarrow B\,x \\ B\,x, B\,y \longrightarrow B\,(x \cap y) \qquad B\,x, B\,y \longrightarrow B\,(x \cup y) \qquad B\,x, B\,y \longrightarrow B\,(x \supset y) \\ \forall u[B\,(X\ u)] \longrightarrow B\,(\forall_i u.Xu) \qquad \forall u[B\,(X\ u)] \longrightarrow B\,(\exists_i u.Xu) \\ B\,I, \Delta \longrightarrow C \end{array}}{form_i(I), \Delta \longrightarrow C}\ form_i\mathcal{L}.$$

In fact, we can consider a more general version of this rule, where classical contexts can be added on both sides of the sequent, like in Proposition 1.

In general, given an object logic L with j connectives $\diamond_j$ of arity greater or equal to zero and a first order quantifier *quant*, the predicate $form_L(\cdot) : bool \to o$ is defined as follows:

$$\begin{array}{lll} form_L(x) \triangleq & atomic(x) & \oplus \\ & \{\exists y_1 \ldots y_n.[x = \diamond_j(y_1, \ldots, y_n) \otimes form_L(y_1) \otimes \ldots \otimes form_L(y_n)]\}_j & \oplus \\ & \exists X.[(x = quant\ u.X\,u) \otimes (\forall u.form_L(X\ u))] & \end{array}$$

It is well known that proving cut-elimination for a logic with definitions *and* induction is not easy [MM00]. The method developed for cut-elimination of *Llinda* (see [Pim05]) is based on some of the ideas present in [Tiu04] and uses a particular notion of *rank* of cut formulas that depends on the level of the formula and on the shape of the derivation itself.

4 Encoding sequent systems

Let *bool* be the type of object-level propositional formulas and let $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ be two meta-level predicates, both of type $bool \to o$.

Consider the object-level sequent $B_1, \ldots, B_n \longrightarrow C_1, \ldots, C_m \quad (n, m \geq 0)$. We encode the sequent above as the linear logic formula: $\lfloor B_1\rfloor \parr \cdots \parr \lfloor B_n\rfloor \parr \lceil C_1\rceil \parr \cdots \parr \lceil C_m\rceil$. The $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ predicates are used in order to identify which object-level formulas appear on which side of the sequent arrow.

Encoding structural rules. The structural rules *weakening* and *contraction* are encoded using the ? of linear logic together by the clauses:

$$\forall B(\lceil B\rceil \circ\!\!- ?\lceil B\rceil) \quad (Neg) \qquad\qquad \forall B(\lfloor B\rfloor \circ\!\!- ?\lfloor B\rfloor) \quad (Pos).$$

Neg and *Pos* will be called *structural clauses*. All object-level two-sided sequents $\Delta \longrightarrow \Gamma$ considered here will be restricted so that Δ and Γ are either multisets or sets of formulas. Sets are used if the structural rules are implicit; multisets are used if no structural rule is implicit. We will assume that exchange is always implicit.

The initial and cut rules. The initial rule, which asserts that the sequent $B \longrightarrow B$ is provable, is represented by the following clause, which has a head with two atoms and no body.

$$\forall B(\lfloor B\rfloor \parr \lceil B\rceil) \qquad\qquad (Init)$$

The cut rule can be specified as following clause with an empty head and two atomic bodies.

$$\forall B(\lceil B\rceil \multimap \lfloor B\rfloor \multimap \bot) \qquad\qquad (Cut)$$

Other variations on the cut rule appear in the literature and many of these can be encoded by changing one or both of the $\multimap$ to $\Rightarrow$. Since the formula *Cut* entails these other variations, so we shall not consider them further.

The *Init* and *Cut* clauses together proves that $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ are duals of each other: that is, they entail the equivalence $\forall B(\lfloor B\rfloor^{\perp} \equiv \lceil B\rceil)$. Notice that this duality of the object-level sequent system becomes a concise equivalence in classical linear logic via negation.

Encoding inference rules. Let $\mathcal{Q}$ be a fixed a set of unary meta-level predicates all of type $bool \to o$. Object-level logical constants will also be assumed to be fixed. These constants will have types of order 0, 1, or 2 and all will build terms of type *bool*. Object-level quantification is first-order and over one domain, denoted at the meta-level by i.

Definition 7. *An* introduction clause *is an uncurried closed flat formula of the form*

$$\forall x_1 \ldots \forall x_n[q(\diamond(x_1, \ldots, x_n)) \circ\!\!- B]$$

where $\diamond$ is an object-level connective of arity n ($n \geq 0$) and q is a meta-level predicate. Furthermore, an atom occurring in B is either of the form $p(x_i)$ or

$p(x_i(y))$ where p is a meta-level predicate and $1 \leq i \leq n$. In the first case, x_i has a type of order 0 while in the second case x_i has a type of order 1 and y is a variable quantified (universally or existentially) in B (in particular, y is not in $\{x_1, \ldots, x_n\}$).

In the inference systems we shall consider now, the set of meta-level predicates $\mathcal{Q}$ is exactly the set $\{\lfloor\cdot\rfloor, \lceil\cdot\rceil\}$. In Section 9, we will consider Girard's **LU** proof system [Gir93] and there we will use some additional meta-level predicates. See [MP04] for other examples of encodings of sequent systems.

5 Introduction clauses as definitions

Given an encoded sequent system $\mathcal{P}$ and an object-level connective $\diamond$ of arity $n \geq 0$, list all the formulas in $\mathcal{P}$ that specify a left-introduction rule for $\diamond$ as:

$$\forall\bar{x}(\lfloor\diamond(x_1, \ldots, x_i)\rfloor \circ\!\!- L_1) \quad \cdots \quad \forall\bar{x}(\lfloor\diamond(x_1, \ldots, x_i)\rfloor \circ\!\!- L_p) \quad (p \geq 0).$$

Similarly, list all the formulas in $\mathcal{P}$ that specify a right-introduction rule for $\diamond$:

$$\forall\bar{x}(\lceil\diamond(x_1, \ldots, x_i)\rceil \circ\!\!- R_1) \quad \cdots \quad \forall\bar{x}(\lceil\diamond(x_1, \ldots, x_i)\rceil \circ\!\!- R_q) \qquad (q \geq 0)$$

All of these $p+q$ displayed formulas can be replaced by the following two clauses

$$\forall\bar{x}(\lfloor\diamond(x_1, \ldots, x_i)\rfloor \circ\!\!- L_1 \oplus \cdots \oplus L_p) \text{ and } \forall\bar{x}(\lceil\diamond(x_1, \ldots, x_i)\rceil \circ\!\!- R_1 \oplus \cdots \oplus R_q)$$

(An empty $\oplus$ is written as the linear logic additive false 0.)

Definition 8. *The formulas*

$$\forall\bar{x}(\lfloor\diamond(x_1, \ldots, x_i)\rfloor \stackrel{\triangle}{=} L_1 \oplus \cdots \oplus L_p) \text{ and } \forall\bar{x}(\lceil\diamond(x_1, \ldots, x_i)\rceil \stackrel{\triangle}{=} R_1 \oplus \cdots \oplus R_q)$$

are said to represent the introduction rules for the object level connective $\diamond$ in their definition form.

Hence introduction clauses of encoded sequent systems form a *flat definition*. As an example, Figures 1 and 2 present the definitions $\mathcal{LK}$ and $\mathcal{LJ}$, respectively. Notice that these specifications are identical except for a systematic renaming of logical constants. To state the formal difference between these two formalisms, we first introduce the named formulas in Figure 3. Notice that the *Cut* and *Init* rules are encoded not as definitions but as formulas.

The following correctness of the LJ and LK encoding is proved in [MP04, Prop 4.2]: The definition $\mathcal{LJ}$ along with the formulas $\{Cut, Init, Pos\}$ correctly represents the provability in LJ, while the definition $\mathcal{LK}$ along with the formulas $\{Cut, Init, Pos, Neg\}$ correctly represents the provability in LK. Thus, LK and LJ are distinquished by specifying whether or not structural rules can be applied to formulas on the right.

An interesting question regarding the formulas appearing in Figure 3 is whether or not the atom-restricted version of each formula entails its general

$$
\begin{array}{llll}
(\Rightarrow L) & \lfloor A \Rightarrow B \rfloor \triangleq \lceil A \rceil \circ\!\!- \lfloor B \rfloor. & (\Rightarrow R) & \lceil A \Rightarrow B \rceil \triangleq \lfloor A \rfloor \otimes \lceil B \rceil. \\
(\wedge L) & \lfloor A \wedge B \rfloor \triangleq \lfloor A \rfloor \oplus \lfloor B \rfloor. & (\wedge R) & \lceil A \wedge B \rceil \triangleq \lceil A \rceil \,\&\, \lceil B \rceil. \\
(\vee R) & \lceil A \vee B \rceil \triangleq \lceil A \rceil \oplus \lceil B \rceil. & (\vee L) & \lfloor A \vee B \rfloor \triangleq \lfloor A \rfloor \,\&\, \lfloor B \rfloor. \\
(\forall_c L) & \lfloor \forall_c B \rfloor \triangleq \lfloor Bx \rfloor. & (\forall_c R) & \lceil \forall_c B \rceil \triangleq \forall x \lceil Bx \rceil. \\
(\exists_c L) & \lfloor \exists_c B \rfloor \triangleq \forall x \lfloor Bx \rfloor. & (\exists_c R) & \lceil \exists_c B \rceil \triangleq \lceil Bx \rceil. \\
(f_c L) & \lfloor f_c \rfloor \triangleq \top. & (t_c R) & \lceil t_c \rceil \triangleq \top.
\end{array}
$$

Fig. 1. Definition $\mathcal{LK}$

$$
\begin{array}{llll}
(\supset L) & \lfloor A \supset B \rfloor \triangleq \lceil A \rceil \circ\!\!- \lfloor B \rfloor. & (\supset R) & \lceil A \supset B \rceil \triangleq \lfloor A \rfloor \otimes \lceil B \rceil. \\
(\cap L) & \lfloor A \cap B \rfloor \triangleq \lfloor A \rfloor \oplus \lfloor B \rfloor. & (\cap R) & \lceil A \cap B \rceil \triangleq \lceil A \rceil \,\&\, \lceil B \rceil. \\
(\cup R) & \lceil A \cup B \rceil \triangleq \lceil A \rceil \oplus \lceil B \rceil. & (\cup L) & \lfloor A \cup B \rfloor \triangleq \lfloor A \rfloor \,\&\, \lfloor B \rfloor. \\
(\forall_i L) & \lfloor \forall_i B \rfloor \triangleq \lfloor Bx \rfloor. & (\forall_i R) & \lceil \forall_i B \rceil \triangleq \forall x \lceil Bx \rceil. \\
(\exists_i L) & \lfloor \exists_i B \rfloor \triangleq \forall x \lfloor Bx \rfloor. & (\exists_i R) & \lceil \exists_i B \rceil \triangleq \lceil Bx \rceil. \\
(f_i L) & \lfloor f_i \rfloor \triangleq \top. & (t_i R) & \lceil t_i \rceil \triangleq \top.
\end{array}
$$

Fig. 2. Definition $\mathcal{LJ}$

$$
\begin{array}{ll}
APos = \forall A(\lfloor A \rfloor \circ\!\!- ?\lfloor A \rfloor \circ\!\!- atomic(A)). & Pos = \forall B(\lfloor B \rfloor \circ\!\!- ?\lfloor B \rfloor). \\
ANeg = \forall A(\lceil A \rceil \circ\!\!- ?\lceil A \rceil \circ\!\!- atomic(A)). & Neg = \forall B(\lceil B \rceil \circ\!\!- ?\lceil B \rceil). \\
AInit = \forall A(\lfloor A \rfloor \otimes \lceil A \rceil \circ\!\!- atomic(A)). & Init = \forall B(\lfloor B \rfloor \otimes \lceil B \rceil). \\
ACut = \forall A(\bot \circ\!\!- \lfloor A \rfloor \circ\!\!- \lceil A \rceil \circ\!\!- atomic(A)). & Cut = \forall B(\bot \circ\!\!- \lfloor B \rfloor \circ\!\!- \lceil B \rceil)
\end{array}
$$

Fig. 3. Some formulas named.

version. Proving the entailment $ACut \vdash Cut$ allows us to conclude that non-atomic cuts can always be reduced to the atomic case. A full cut-elimination proof then only needs to deal with eliminating atomic cuts. Section 7 provides conditions on inference rule encodings that ensures that this entailment can be proved. Dually, the entailment $AInit \vdash Init$ allows us to eliminate non-atomic initial rules, a property that helps can be used to judge the design of a good proof system, especially when using *synthetic connectives* (see [Gir99]). Elimination of non-atomic initial rules is discussed further in Section 8. Finally, it is worthy to say that restricting logical rules and axioms to the atomic case also plays a central role in *Calculus of Structures* [Gug05].

6 Bipolar clauses

In this section we shall clarify better the role of bipolar clauses in the specification of sequent systems.

Since introduction clauses are defined as flat clauses, they are bipolar. It is interesting to ask, however, if there exist sequent calculus inference rules that can be encoded in linear logic by a formula that is not necessarily bipolar.

Suppose that c is the introduction clause $\forall \bar{x}.[q(\diamond(x_1,\ldots,x_n)) \stackrel{\triangle}{=} B]$ corresponds to a sequent calculus specification. This means that, when doing some meta-level reasoning, backchaining over c:

$$\frac{\begin{array}{c}\Pi' \\ \Delta \longrightarrow \Gamma, B\bar{t}\end{array}}{\Delta \longrightarrow q(\diamond(t_1,\ldots,t_n)), \Gamma}\ def\mathcal{R}$$

must mimic exactly the behavior of the inference rule for $\diamond$. Hence the body B *must be* decomposed at once before some other meta level action can be done. That is, B cannot interact with any possible context in Π'. The *focussing* property of linear logic guarantees this only if no synchronous connective is in the scope of an asynchronous connective; that is, if c is bipolar.

Example 1. Consider the following clauses:

$$\lceil\diamond(A,B,C)\rceil \circ\!\!- \lceil A\rceil \,\&\, (\lceil B\rceil \otimes \lceil C\rceil) \qquad \lfloor\diamond(A,B,C)\rfloor \circ\!\!- \lfloor A\rfloor \oplus (\lfloor B\rfloor \parr \lfloor C\rfloor)$$

Note that the first clause is not bipolar. If they are to correspond to the encoding of sequent inference rules, the *natural* candidates would be

$$\frac{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, A \quad \Gamma_1 \vdash \Delta_1, B \quad \Gamma_2 \vdash \Delta_2, C}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, \diamond(A,B,C)}$$

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, \diamond(A,B,C) \vdash \Delta} \qquad \frac{\Gamma, B, C \vdash \Delta}{\Gamma, \diamond(A,B,C) \vdash \Delta}$$

But it turns out that while at the meta level it is possible to prove the sequent $!\mathit{Init} \vdash \lceil A\rceil \,\&\, (\lceil B\rceil \otimes \lceil C\rceil), \lfloor A\rfloor \oplus (\lfloor B\rfloor \parr \lfloor C\rfloor)$ at the object level the two sequent rules listed above cannot be used to prove $\diamond(A,B,C) \vdash \diamond(A,B,C)$. That is, this object-logic sequent can be proved only a non-atomic instance of the initial rule. Hence, *provability* is not the same and the flat clauses above are not *adequate* for representing the inference figures.

Once we know that introduction clauses must necessarily be bipolar, the next question that arises is if every introduction clause is a meta-level representation of a sequent inference rule. This can be shown by a straightforward case analysis.

Proposition 3. *Every introduction clause corresponds to a specification of a sequent calculus introduction rule.*

7 Canonical and coherent proof systems

The purpose of strengthening linear logic with definitions and induction is to enhance the number of properties about encoded proof systems that can be formally proved inside the framework. In this section we will present a necessary condition for characterizing systems having the cut-elimination property.

Definition 9. *A* canonical proof system *is a set $\mathcal{P}$ of flat clauses such that (i) the initial clause is a member of $\mathcal{P}$, (ii) the cut clause is a member of $\mathcal{P}$, (iii) structural clauses (Pos and Neg) may be members of $\mathcal{P}$, and (iv) all other clauses in $\mathcal{P}$ are introduction clauses with the additional restriction that, for every pair of atoms of the form $\lfloor T \rfloor$ and $\lceil S \rceil$ in a body, the head variable of T differs from head variable of S. A formula that satisfies condition (iv) is also called a* canonical clause.

Definition 10. *Consider a canonical proof system $\mathcal{P}$ and an object-level connective, say, $\diamond$ of arity $n \geq 0$. Let the formulas*

$$\forall \bar{x}(\lfloor \diamond(x_1, \ldots, x_n) \rfloor \triangleq B_l) \quad \textit{and} \quad \forall \bar{x}(\lceil \diamond(x_1, \ldots, x_n) \rceil \triangleq B_r)$$

be the definition form for the left and right introduction rules for $\diamond$. The object-level connective $\diamond$ has dual left and right introduction rules *if $!\,Cut \vdash \forall \bar{x}(B_l \multimap B_r \multimap \bot)$ in linear logic.*

Definition 11. *A canonical system is called* coherent *if the left and right introduction rules for each object-level connective are duals.*

The cut-elimination theorem for a particular logic can often be divided into two parts. The first part shows that a cut involving a non-atomic formula can be replaced by possibly multiple cuts involving subformulas of the original cut formula. This process stop when cut formulas are atoms. This part of the proof works because left and right introduction rules for each logical connective are duals (formalized here in Definition 10). The second part of the proof argues how cuts with atomic formulas can be removed. Cut-elimination for coherent proof systems is proved similarly: Theorem 1 shows that non-atomic cuts can be reduced to atomic. The remarkable aspect about this is that this part of the cut-elimination process is done *entirely* inside the logical framework *Llinda*.

Proving that atomic cuts can be eliminated requires induction over proofs, hence the reasoning cannot be done inside *Llinda*. This was done in [MP02], where it was also shown that "being coherent" is a general and decidable characterization. Since all the reasoning is done using linear logic, the essence of cut-elimination can be captured totally at the meta-level. Hence it is, in fact, independent of the object logic analyzed.

Theorem 1. *Let $\mathcal{P}$ be a coherent system and let form$(\cdot)$ be the inductive predicate defining object-level formulas. The sequent*

$$\mathcal{P} \parallel !\,Init, !\,ACut, form(B) \longrightarrow Cut(B)$$

is provable in Llinda.

Proof The proof is by induction where the invariance is $\lambda x.\, !\,Cut(x)$. Consider the following derivation

$$\dfrac{\dfrac{!\,Cut(x_1), \ldots, !\,Cut(x_n), B_r, B_l \longrightarrow \bot}{!\,Cut(x_1), \ldots, !\,Cut(x_n), \lfloor \diamond(x_1, \ldots, x_n) \rfloor, \lceil \diamond(x_1, \ldots, x_n) \rceil \longrightarrow \bot}\, def\mathcal{L}}{!\,Cut(x_1), \ldots, !\,Cut(x_n) \longrightarrow !\,Cut(\diamond(x_1, \ldots, x_n))}\, !\,R, \multimap R$$

By definition of coherent systems, the sequent

$$!\,Cut(x_1), \ldots, !\,Cut(x_n), B_r, B_l \longrightarrow \bot$$

is provable. The second part is trivial and consists on proving the sequent $\lambda x.!\,Cut(x), !\,ACut, atomic(B) \longrightarrow Cut(B)$. ∎

8 Homogeneous systems

Theorem 1 shows that, for coherent systems, the cut rule can be restricted to the atomic case. A similar problem is that of analyzing when the initial rule can be also restricted to the atomic case. It turns out that duality is not enough for this case.

Example 2. Consider the connective $\diamond(A, B, C)$ with associated rules:

$$\frac{\Gamma \vdash \Delta, A}{\Gamma \vdash \Delta, \diamond(A,B,C)}\ (\diamond R_1) \qquad \frac{\Gamma \vdash \Delta, B \quad \Gamma \vdash \Delta, C}{\Gamma \vdash \diamond(A,B,C)}\ (\diamond R_2)$$

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, \diamond(A,B,C) \vdash \Delta}\ (\diamond L_1) \qquad \frac{\Gamma, A \vdash \Delta \quad \Gamma, C \vdash \Delta}{\Gamma, \diamond(A,B,C) \vdash \Delta}\ (\diamond L_2)$$

These rules can be specified in *Llinda* as:

$$\lceil \diamond(A,B,C) \rceil \triangleq \lceil A \rceil \oplus (\lceil B \rceil \,\&\, \lceil C \rceil) \quad \lfloor \diamond(A,B,C) \rfloor \triangleq (\lfloor A \rfloor \,\&\, \lfloor B \rfloor) \oplus (\lfloor A \rfloor \,\&\, \lfloor C \rfloor)$$

It is easy to see that $!\,Cut, \lceil A \rceil \oplus (\lceil B \rceil \,\&\, \lceil C \rceil), (\lfloor A \rfloor \,\&\, \lfloor B \rfloor) \oplus (\lfloor A \rfloor \,\&\, \lfloor C \rfloor) \vdash \bot$ holds and hence a system formed with these two defined rules plus *initial* and *cut* is coherent.

However, the sequent $!\,Init \vdash \lceil A \rceil \oplus (\lceil B \rceil \,\&\, \lceil C \rceil), (\lfloor A \rfloor \,\&\, \lfloor B \rfloor) \oplus (\lfloor A \rfloor \,\&\, \lfloor C \rfloor)$ is not provable. This reflects the fact that, at the object-level, the sequent $\diamond(A,B,C) \longrightarrow \diamond(A,B,C)$ has only the trivial proof: the one where the only rule applied is the initial rule. The formula $\diamond(A,B,C)$, hence, cannot be decomposed.

The problem with the system above is that the introduction rules for the connective $\diamond$ are not *homogeneous*, in the sense that their meta-level behavior is captured using connectives of different polarities[2] (see [Gir99] for an object-level discussion on *syntectic connectives*).

Definition 12. *A coherent system is* homogeneous *if all connectives appearing in a body of a defined rule have the same polarity.*

Theorem 2. *Let $\mathcal{P}$ be a coherent system and let* form$(\cdot)$ *be the inductive predicate for object-level formulas. If $\mathcal{P}$ is homogeneous then the following is provable.*

$$\mathcal{P} \parallel !\,AInit, form(B) \longrightarrow Init(B)$$

Proof Let $\mathcal{P}$ be a homogeneous system. Since $\mathcal{P}$ is coherent, it is easy to see that all left and right bodies of defined clauses are dual linear logic formulas. Hence the result follows by structural induction (invariant $\lambda x.!\,Init(x)$). ∎

[2] Note that, as Example 2 shows, at the meta-level the encoding of dual rules may not be dual linear logic formulas.

9 LU

In [Gir93], Girard introduced the sequent system **LU** (logic of unity) in which classical, intuitionistic, and linear logics appear as fragments. In this logic, all three of these logics keep their own characteristics but they can also communicate via formulas containing connectives mixing these logics. The key to allowing these logics to share one proof system lies in using *polarities*. In terms of the encoding we have presented here, polarities allow the meta-level atom $\lfloor B \rfloor$ be replaced by $?\lfloor B \rfloor$ if B is positive and the meta-level atom $\lceil B \rceil$ be replaced by $?\lceil B \rceil$ if B is negative. This possibility of replacement is in contrast to the examples of classical and intuitionistic sequent proof systems presented earlier where $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ atoms are either all preceded by the ? modal or all are not so prefixed. The neutral polarity is also available and corresponds to the case where this replacement with a ? modal is not allowed. Many of the **LU** inference rules for classical and intuitionistic connectives are specified in Figure 4. The definition of the predicates $pos(\cdot)$, $neg(\cdot)$, and $neu(\cdot)$ can be directly obtained from the various polarity tables given in [Gir93]. These definitions, together with the ones in Figure 5 will be denoted by $\mathcal{P}$.

Proving cut-elimination for **LU** is not at all easy: there are some rules concerning polarities that have an empty head and bodies with an erase function. In this particular case, moving the cut up is not possible for some proofs, and the usual cut-elimination proof doesn't work.

LU is not canonical since the side conditions in its rules require meta-level predicates other than simply $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$. On proving cut-elimination for coherent systems, it was essential to restrict the predicates to left and right since the cut rule is a rule about duality of these two predicates. In the case of allowing some other predicates one have to be careful on reasoning about proofs where rules concerning these predicates are applied.

Proposition 4. *The following clauses can be proved in Llinda*

$$\forall B.(pos(B) \Rightarrow neg(B) \Rightarrow 0). \quad \forall B.(pos(B) \Rightarrow neu(B) \Rightarrow 0).$$
$$\forall B.(neg(B) \Rightarrow neu(B) \Rightarrow 0).$$

These clauses play the role of the *Cut* rule on determining the dual predicates for polarities. Let $\mathcal{L}$ be the set of clauses above. The following is easily proved (the proof can be automated in the same way as described in [MP02]).

Proposition 5. *For every connective $\diamond$ of **LU**, if the left and right introduction clauses for $\diamond$ in their definition form are $\forall \bar{x}(\lfloor \diamond(x_1, \ldots, x_i) \rfloor \circ\!\!- B_l)$ and $\forall \bar{x}(\lceil \diamond(x_1, \ldots, x_i) \rceil \circ\!\!- B_r)$ then*

$$\mathcal{P} \parallel !\,\mathcal{L}, !\,Init, !\,Cut, !\,Pos, !\,Neg \vdash \forall \bar{x}(B_l \multimap B_r \multimap \bot) \qquad (1)$$

in linear logic, where Neg is the third and Pos the fourth clause in Figure 4.

This suggests that such an entailment might be used as a natural generalization of coherence to this setting. In fact, we have the following results:

Identity and structure

$$\lfloor B\rfloor \parr \lceil B\rceil. \qquad \bot \circ\!\!- \lfloor B\rfloor \circ\!\!- \lceil B\rceil.$$
$$\lceil N\rceil \circ\!\!- ?\lceil N\rceil \Leftarrow neg(N). \qquad \lfloor P\rfloor \circ\!\!- ?\lfloor P\rfloor \Leftarrow pos(P).$$

Conjunction

$$\lceil A \wedge B\rceil \triangleq\ !\lceil A\rceil \otimes\ !\lceil B\rceil \otimes !(pos(A) \oplus pos(B)). \qquad \lceil A \wedge B\rceil \triangleq \lceil A\rceil \,\&\, \lceil B\rceil \otimes !(notpos(A)\ \&\ notpos(B)).$$
$$\lfloor A \wedge B\rfloor \triangleq\ ?\lfloor A\rfloor \parr ?\lfloor B\rfloor \otimes !(pos(A) \oplus pos(B)). \qquad \lfloor A \wedge B\rfloor \triangleq \lfloor A\rfloor \oplus \lfloor B\rfloor \otimes !(notpos(A)\ \&\ notpos(B)).$$

Intuitionistic implication

$$\lceil A \supset B\rceil \triangleq\ ?\lfloor A\rfloor \parr \lceil B\rceil. \qquad \lfloor A \supset B\rfloor \triangleq\ !\lceil A\rceil \otimes \lfloor B\rfloor.$$

Quantifiers

$$\lceil \forall A\rceil \triangleq \forall x\, ?\lceil Ax\rceil. \qquad \lfloor \forall A\rfloor \triangleq\ !\lfloor Ax\rfloor.$$
$$\lceil \exists A\rceil \triangleq\ !\lceil Ax\rceil. \qquad \lfloor \exists A\rfloor \triangleq \forall x\, ?\lfloor Ax\rfloor.$$

Disjunction

$$\lceil A \vee B\rceil \triangleq\ !\lceil A\rceil \oplus\ !\lceil B\rceil \otimes !(notneg(A)\ \&\ notneg(B)).$$
$$\lceil A \vee B\rceil \triangleq\ ?\lceil A\rceil \parr ?\lceil B\rceil \otimes !((pos(A)\ \&\ neg(B)) \oplus (neg(A)\ \&\ notneu(B))).$$
$$\lceil A \vee B\rceil \triangleq \lceil A\rceil \parr ?\,!\lceil B\rceil \otimes !(neg(A)\ \&\ neu(B)).$$
$$\lceil A \vee B\rceil \triangleq\ ?\,!\lceil A\rceil \parr \lceil B\rceil \otimes !(neu(A)\ \&\ neg(B)).$$
$$\lfloor A \vee B\rfloor \triangleq\ ?\lfloor A\rfloor \,\&\, ?\lfloor B\rfloor \otimes !(notneg(A)\ \&\ notneg(B)).$$
$$\lfloor A \vee B\rfloor \triangleq\ !\lfloor A\rfloor \otimes\ !\lfloor B\rfloor \otimes !((pos(A)\ \&\ neg(B)) \oplus (neg(A)\ \&\ notneu(B))).$$
$$\lfloor A \vee B\rfloor \triangleq \lfloor A\rfloor \otimes\ !\,?\lfloor B\rfloor \otimes !(neg(A)\ \&\ neu(B)).$$
$$\lfloor A \vee B\rfloor \triangleq\ !\,?\lfloor A\rfloor \otimes \lfloor B\rfloor \otimes !(neu(A)\ \&\ neg(B)).$$

Classical implication

$$\lceil A \Rightarrow B\rceil \triangleq\ ?\lfloor A\rfloor \parr ?\lceil B\rceil \otimes !((neg(A)\ \&\ neg(B)) \oplus (pos(A)\ \&\ notneu(B))).$$
$$\lceil A \Rightarrow B\rceil \triangleq \lceil B\rceil \oplus \lfloor A\rfloor \otimes !(neg(A)\ \&\ pos(B)).$$
$$\lfloor A \Rightarrow B\rfloor \triangleq \lceil A\rceil \,\&\, \lfloor B\rfloor \otimes !(neg(A)\ \&\ pos(B)).$$
$$\lfloor A \Rightarrow B\rfloor \triangleq\ !\lceil A\rceil \otimes\ !\lfloor B\rfloor \otimes !((neg(A)\ \&\ neg(B)) \oplus (pos(A)\ \&\ notneu(B))).$$

Fig. 4. LU rules

$$notpos(A) \triangleq (neu(A) \oplus neg(A)). \qquad notneg(A) \triangleq (neu(A) \oplus pos(A)).$$
$$notneu(A) \triangleq (neg(A) \oplus pos(A)).$$

Fig. 5. Polarities

Theorem 3. *Let* $\boldsymbol{LU}$ *be the encoding for LU (including the polarity table and definitions in Figure 5). The following is provable in* Linda*:*

$$\boldsymbol{LU} \parallel\ !\,\mathcal{L}, !\,Init, !\,APos, !\,ANeg, !\,ACut \rightarrow Pos \otimes Neg \otimes Cut.$$

Theorem 4. *Let* B *be the encoding of an object-level LU formula. If*

$$\boldsymbol{LU} \parallel\ !\,\mathcal{L}, !\,Init, !\,Cut, !\,Pos, !\,Neg \rightarrow B$$

is provable then there is a proof of the same sequent without backchaining over the Cut clause.

10 Conclusion

We have illustrated how object-level sequent calculus proof systems can be encoded into linear logic in such a way that the meta-theory of linear logic helps to

establish formal meta-theoretic properties of the object-logic proof system. By strengthening linear logic with a form of induction, much of this meta-theory can be captured entirely in the meta-logic. We illustrated our approach by showing how such a meta-level approach can be used to establish cut-elimination for LU.

References

[And92] J.-M. Andreoli. Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation*, 2(3):297–347, 1992.

[Chu40] A. Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[FM88] A. Felty and D. Miller Specifying theorem provers in a higher-order logic programming language, *Ninth International Conference on Automated Deduction*, 1988.

[Gen69] G. Gentzen. Investigations into logical deductions. In M. E. Szabo, editor, *The Collected Papers of Gerhard Gentzen*, pp. 68–131. North-Holland Publishing Co., Amsterdam, 1969.

[Gir87] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, vol. 50, pp. 1-102, 1987.

[Gir93] J.-Y. Girard. On the unity of logic. *Ann. of Pure and Applied Logic*, 59:201–217, 1993.

[Gir99] J.-Y. Girard. On the meaning of logical rules I: syntax vs. semantics. *Computational Logic*, eds Berger and Schwichtenberg, pp. 215-272, SV, 1999.

[Gug05] A. Guglielmi A system of Interaction and Structure. *ACM Transactions in Computational Logic*, to appear, 2005.

[Har93] R. Harper, F. Honsell, and G. Plotkin A framework for defining logics, *Journal of the ACM*, vol.40(1), pp. 143-184, 1993.

[Mil96] Dale Miller. Forum: A multiple-conclusion specification language. *Theoretical Computer Science*, 165(1):201–232, September 1996.

[MM00] R. McDowell and D. Miller. Cut-elimination for a logic with definitions and induction. *Theoretical Computer Science*, 232:91–119, 2000.

[MP04] D. Miller and E. Pimentel. Linear logic as a framework for specifying sequent calculus. *Lecture Notes in Logic 17, Logic Colloquium'99*, 2004.

[MP02] D. Miller and E. Pimentel. Using linear logic to reason about sequent systems. *Proceedings of Tableaux 2002*, LNAI 2381, 2002.

[NM88] G. Nadathur and D. Miller. An Overview of λProlog. In *Fifth International Logic Programming Conference*, pp. 810–827, August 1988. MIT Press.

[Pfn89] F. Pfenning. Elf: A Language for Logic Definition and Verified Metaprogramming. *Fourth Annual Symposium on Logic in Computer Science*, 1989.

[Pfn95] F. Pfenning. Structural Cut Elimination. *Proceedings, Tenth Annual IEEE Symposium on Logic in Computer Science*, 1995.

[Pfn00] F. Pfenning. Structural Cut Elimination: I. Intuitionistic and Classical Logic. *Information and Computation*, 157(1-2) pp. 84-141, 2000.

[Pim01] E. G. Pimentel. *Lógica linear e a especificação de sistemas computacionais*. PhD thesis, Universidade Federal de Minas Gerais, Belo Horizonte, M.G., Brasil, December 2001. (written in English).

[Pim05] E. G. Pimentel. *Cut elimination for Llinda*, 2005. Draft available from http://www.mat.ufmg.br/~elaine

[Tiu04] A. Tiu. *A Logical Framework for Reasoning about Logical Specifications*. PhD thesis, Penn State University, 2004.