LINEAR LOGIC AS A FRAMEWORK FOR SPECIFYING SEQUENT CALCULUS

DALE MILLER AND ELAINE PIMENTEL

Abstract. In recent years, intuitionistic logic and type systems have been used in numerous computational systems as frameworks for the specification of natural deduction proof systems. As we shall illustrate here, linear logic can be used similarly to specify the more general setting of sequent calculus proof systems. Linear logic's meta theory can be used also to analyze properties of a specified object-level proof system. We shall present several example encodings of sequent calculus proof systems using the Forum presentation of linear logic. Since the object-level encodings result in logic programs (in the sense of Forum), various aspects of object-level proof systems can be automated.

§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 generally been based on intuitionistic logic in which quantification at (non-predicate) higher-order types is available. Identifying a framework that allows the specification of a wide range of logics has proved to be most practical since a single implementation of such a framework can then be used to provide various degrees of automation of object-logics. For example, Isabelle [26] and λProlog [25] are implementations of an intuitionistic logic subset of Church's Simple Theory of Types, while Elf [27] is an implementation of a dependently typed λ-calculus [16]. These computer systems have been used as meta languages to automate various aspects of various 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, if the meta-logic is rich enough to include λ-bindings in its syntax and to provide α and β conversion as part of its equality of syntax (as is the case for the systems mentioned above), the object-logics immediately inherit such simple and declarative treatments of binding constructs and substitutions. 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 making it difficult to have natural encodings of any logic for which these structural rules are not intended. Also, intuitionistic logic does not have an involutive negation, making it difficult to address directly dualities in object-logics.

In this paper, we make use of linear logic as a meta-logic and find that we can specify a variety of proof systems for object-level systems. By making use of classical linear logic, we are able to capture not only natural deduction proof systems but also many sequent calculus proof systems. We will present our scheme for encoding proof systems in linear logic and show several examples of making such specifications. Since the encodings of such logical systems are natural and direct the rich meta-theory of linear logic can be used to drawing conclusions about the object-level proof systems, and we illustrate such reasoning as well.

This paper is organized as follows: Section 2 gives an introduction to linear logic and Forum. Section 3 shows the representation of sequents and inference rules, while Forum encodings of the well-known proof systems for linear, classical and intuitionistic logics are presented in Section 4. Using the meta-theory, it is possible to prove the collapsing of some modal prefixes for the specified classical and intuitionistic systems. In Section 5 other sequent calculus for these logics are encoded where modal prefixes collapse less dramatically. In order to show how to represent systems that make use of polarities, Section 6 presents an encoding of the so called Logic of Unity (LU) proof system. Section 7 provides an overview of how one might proof search for both Forum and encoded object-level proof systems. We conclude and discuss some future research directions in Section 8.

Our main purpose in this paper is to illustrate via examples how linear logic can be used to both specify and reason about object-level sequent proof systems. We shall do this largely by presenting a series of examples. More extensive discussion of the material in this paper can be found in the PhD dissertation of the second author [28].

§2. Overview of Linear Logic and Forum. Linear Logic [13] uses the following logical connectives: the exponentials ! and ?; $\otimes$, $⅋$, $\bot$, and 1 for the multiplicative conjunction, disjunction, false, and true; &, $\oplus$, 0, $\top$ for the additive version of these connectives; $\multimap$ for linear implication, and $\forall$ and $\exists$ for universal and existential quantification. We shall assume that the reader is familiar with the sequent calculus presentation of linear logic and with its basic properties.

2.1. The Forum presentation of linear logic. The connectives of linear logic can be classified as *synchronous* and *asynchronous* [2] depending on whether or not the right introduction rule for that connective needs to "synchronize" with its surrounding context. The de Morgan dual of a connective in one of these classes yields a connective in the other class. Given this division of connectives, Miller proposed in [23] the *Forum* presentation of linear logic in which formulas are build using only the asynchronous connectives, namely, $?$, $\bindnasrepma$, $\bot$, $\&$, $\top$, $\multimap$, and $\forall$, along with the intuitionistic version of implication ($B \Rightarrow C$ denotes $!B \multimap C$). The synchronous connectives are implicitly available in Forum since a two sided sequent is used: connectives appearing on the left of the sequent arrow behave synchronously. Proof search in the Forum presentation of linear logic resembles the search involved in logic programming [24, 22]: introducing asynchronous connectives corresponds to goal-directed search and introducing synchronous connectives corresponds to backchaining over logic program clauses.

Forum is a presentation of all of linear logic since it contains a complete set of connectives. The connectives missing from Forum are directly definable using the following logical equivalences:

$$B^\bot \equiv B \multimap \bot \qquad 0 \equiv \top \multimap \bot \qquad 1 \equiv \bot \multimap \bot \qquad \exists x.B \equiv (\forall x.B^\bot)^\bot$$
$$!B \equiv (B \Rightarrow \bot) \multimap \bot \qquad B \oplus C \equiv (B^\bot \mathbin{\&} C^\bot)^\bot \qquad B \otimes C \equiv (B^\bot \bindnasrepma C^\bot)^\bot$$

The collection of connectives in Forum is not minimal. For example, $?$ and $\bindnasrepma$, can be defined in terms of the remaining connectives:

$$?B \equiv (B \multimap \bot) \Rightarrow \bot \quad \text{and} \quad B \bindnasrepma C \equiv (B \multimap \bot) \multimap C.$$

Here, the equivalence $B \equiv C$ means that the universal closure of the expression $(B \multimap C) \mathbin{\&} (C \multimap B)$ is provable in linear logic.

To help make the connection between proof search in Forum and logic programming, it is useful to introduce the notions of goal and clause into Forum. A formula is a *Forum clause* if it is of the form

$$\forall \bar{y}(G_1 \hookrightarrow \cdots \hookrightarrow G_m \hookrightarrow G_0), \quad (m \geq 0)$$

where $G_0, \ldots, G_m$ are arbitrary Forum formulas and occurrences of $\hookrightarrow$ are either occurrences of $\multimap$ or $\Rightarrow$. A formula of Forum is a *flat goal* if it does not contain occurrences of $\multimap$ and $\Rightarrow$ and all occurrences of $?$ have atomic scope. A Forum clause is a *flat clause* if $G_0, \ldots, G_m$ are flat goals. It is possible to also add the restriction that the formula G_0, the *head* of the clause, is of the form $B_1 \bindnasrepma \cdots \bindnasrepma B_n$, where $n \geq 0$ and each B_i is an atom. If $n = 0$ then we write the head as simply $\bot$ and say that the head is *empty*. A flat clause is essentially a clause of the LinLog system [2] except that heads of flat clauses may be empty. It will be the case that all formulas used to specify sequent calculus inferences rules in this paper will be flat Forum clauses.

As in Church's Simple Theory of Types [6], both terms and formulas are built using a simply typed λ-calculus. 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 $(\lambda x.B)t$. Following [6], we shall also assume that formulas of Forum have type o.

2.2. Proof system for Forum. The proof system for Forum, $\mathcal{F}$, is given in Figure 1. Sequents in $\mathcal{F}$ have the form

$$\Sigma\colon \Psi; \Delta \longrightarrow \Gamma; \Upsilon \quad \text{and} \quad \Sigma\colon \Psi; \Delta \xrightarrow{B} \Gamma; \Upsilon,$$

where Σ is a signature, Δ and Γ are multisets of formulas, Ψ and Υ are sets of formulas, and B is a formula. All formulas in sequents are composed of the asynchronous connectives listed above (together with $\Rightarrow$) and contain at most the non-logical symbols present in Σ (such formulas are called Σ*-formulas*). The intended meanings of these two sequents in linear logic are

$$!\,\Psi, \Delta \longrightarrow \Gamma, ?\,\Upsilon \quad \text{and} \quad !\,\Psi, \Delta, B \longrightarrow \Gamma, ?\,\Upsilon,$$

respectively. In the proof system of Figure 1, the only right rules are those for sequents of the form $\Sigma\colon \Psi; \Delta \longrightarrow \Gamma; \Upsilon$. The syntactic variable $\mathcal{A}$ in Figure 1 denotes a multiset of atomic formulas. Left rules are applied only to the formula B that labels the sequent arrow in $\Sigma\colon \Psi; \Delta \xrightarrow{B} \mathcal{A}; \Upsilon$.

We use the turnstile symbol as the mathematical-level judgment that a sequent is provable: that is, $\Delta \vdash \Gamma$ means that the two-sided sequent $\Delta \longrightarrow \Gamma$ has a linear logic proof. The following correctness theorem for $\mathcal{F}$ is given in [23] and is based on the focusing result of Andreoli in [2].

THEOREM 2.1. *Let Σ be a signature, Δ and Γ be multisets of Σ-formulas, and Ψ and Υ be sets of Σ-formulas. The sequent $\Sigma\colon \Psi; \Delta \longrightarrow \Gamma; \Upsilon$ has a proof in $\mathcal{F}$ if and only if $!\,\Psi, \Delta \vdash \Gamma, ?\,\Upsilon$.*

We shall use the term *backchaining* to refer to an application of either the *decide* or the *decide*! inference rule followed by a series of applications of left-introduction rules (reading a proof bottom-up). This notion of backchaining generalizes the usual notion found in the logic programming literature.

When presenting examples of Forum code we often use $\circ\!-$ and $\Rightarrow$ to be the converses of $\multimap$ and $\Leftarrow$ since they provide a more natural operational reading of clauses (similar to the use of `:-` in Prolog). We will assume that when parsing expressions, $⅋$ and & bind tighter than $\circ\!-$ and $\Leftarrow$.

Multiset rewriting can be captured naturally in proof search. Consider, for example, the clause

$$a ⅋ b \circ\!- c ⅋ d ⅋ e.$$

and the sequent $\Sigma\colon \Psi; \Delta \longrightarrow a, b, \Gamma; \Upsilon$, where the clause displayed above is a member of Ψ. A proof for this sequent can then end with the following inference

$$\frac{}{\Sigma\colon \Psi;\Delta \longrightarrow \top,\Gamma;\Upsilon}\ \top R$$

$$\frac{\Sigma\colon \Psi;\Delta \longrightarrow B,\Gamma;\Upsilon \quad \Sigma\colon \Psi;\Delta \longrightarrow C,\Gamma;\Upsilon}{\Sigma\colon \Psi;\Delta \longrightarrow B \mathbin{\&} C,\Gamma;\Upsilon}\ \& R$$

$$\frac{\Sigma\colon \Psi;\Delta \longrightarrow \Gamma;\Upsilon}{\Sigma\colon \Psi;\Delta \longrightarrow \bot,\Gamma;\Upsilon}\ \bot R \qquad \frac{\Sigma\colon \Psi;\Delta \longrightarrow B,C,\Gamma;\Upsilon}{\Sigma\colon \Psi;\Delta \longrightarrow B \parr C,\Gamma;\Upsilon}\ \parr R$$

$$\frac{\Sigma\colon \Psi;B,\Delta \longrightarrow C,\Gamma;\Upsilon}{\Sigma\colon \Psi;\Delta \longrightarrow B \multimap C,\Gamma;\Upsilon}\ \multimap R \qquad \frac{\Sigma\colon B,\Psi;\Delta \longrightarrow C,\Gamma;\Upsilon}{\Sigma\colon \Psi;\Delta \longrightarrow B \Rightarrow C,\Gamma;\Upsilon}\ \Rightarrow R$$

$$\frac{y:\tau,\Sigma\colon \Psi;\Delta \longrightarrow B[y/x],\Gamma;\Upsilon}{\Sigma\colon \Psi;\Delta \longrightarrow \forall_\tau x.B,\Gamma;\Upsilon}\ \forall R \qquad \frac{\Sigma\colon \Psi;\Delta \longrightarrow \Gamma;B,\Upsilon}{\Sigma\colon \Psi;\Delta \longrightarrow ?B,\Gamma;\Upsilon}\ ?R$$

$$\frac{\Sigma\colon B,\Psi;\Delta \xrightarrow{B} \mathcal{A};\Upsilon}{\Sigma\colon B,\Psi;\Delta \longrightarrow \mathcal{A};\Upsilon}\ \mathit{decide}\,! \qquad \frac{\Sigma\colon \Psi;\Delta \longrightarrow \mathcal{A},B;B,\Upsilon}{\Sigma\colon \Psi;\Delta \longrightarrow \mathcal{A};B,\Upsilon}\ \mathit{decide}\,?$$

$$\frac{\Sigma\colon \Psi;\Delta \xrightarrow{B} \mathcal{A};\Upsilon}{\Sigma\colon \Psi;B,\Delta \longrightarrow \mathcal{A};\Upsilon}\ \mathit{decide}$$

$$\frac{}{\Sigma\colon \Psi;\cdot \xrightarrow{A} A;\Upsilon}\ \mathit{initial} \qquad \frac{}{\Sigma\colon \Psi;\cdot \xrightarrow{A} \cdot;A,\Upsilon}\ \mathit{initial}?$$

$$\frac{}{\Sigma\colon \Psi;\cdot \xrightarrow{\bot} \cdot;\Upsilon}\ \bot L \qquad \frac{\Sigma\colon \Psi;\Delta \xrightarrow{B_i} \mathcal{A};\Upsilon}{\Sigma\colon \Psi;\Delta \xrightarrow{B_1 \& B_2} \mathcal{A};\Upsilon}\ \& L_i \qquad \frac{\Sigma\colon \Psi;B \longrightarrow \cdot;\Upsilon}{\Sigma\colon \Psi;\cdot \xrightarrow{?B} \cdot;\Upsilon}\ ?\mathrm{L}$$

$$\frac{\Sigma\colon \Psi;\Delta_1 \xrightarrow{B} \mathcal{A}_1;\Upsilon \quad \Sigma\colon \Psi;\Delta_2 \xrightarrow{C} \mathcal{A}_2;\Upsilon}{\Sigma\colon \Psi;\Delta_1,\Delta_2 \xrightarrow{B \parr C} \mathcal{A}_1,\mathcal{A}_2;\Upsilon}\ \parr L \qquad \frac{\Sigma\colon \Psi;\Delta \xrightarrow{B[t/x]} \mathcal{A};\Upsilon}{\Sigma\colon \Psi;\Delta \xrightarrow{\forall_\tau x.B} \mathcal{A};\Upsilon}\ \forall L$$

$$\frac{\Sigma\colon \Psi;\Delta_1 \longrightarrow \mathcal{A}_1,B;\Upsilon \quad \Sigma\colon \Psi;\Delta_2 \xrightarrow{C} \mathcal{A}_2;\Upsilon}{\Sigma\colon \Psi;\Delta_1,\Delta_2 \xrightarrow{B \multimap C} \mathcal{A}_1,\mathcal{A}_2;\Upsilon}\ \multimap L$$

$$\frac{\Sigma\colon \Psi;\cdot \longrightarrow B;\Upsilon \quad \Sigma\colon \Psi;\Delta \xrightarrow{C} \mathcal{A};\Upsilon}{\Sigma\colon \Psi;\Delta \xrightarrow{B \Rightarrow C} \mathcal{A};\Upsilon}\ \Rightarrow L$$

FIGURE 1. The $\mathcal{F}$ proof system. The rule ∀R has the proviso that y is not declared in the signature Σ, and the rule ∀L has the proviso that t is a Σ-term of type τ. In $\&L_i$, $i = 1$ or $i = 2$.

rules.

$$\dfrac{\dfrac{\dfrac{\dfrac{\Sigma\colon \Psi;\Delta \longrightarrow c,d,e,\Gamma;\Upsilon}{\Sigma\colon \Psi;\Delta \longrightarrow c,d \parr e,\Gamma;\Upsilon}}{\Sigma\colon \Psi;\Delta \longrightarrow c \parr d \parr e,\Gamma;\Upsilon} \quad \dfrac{\dfrac{}{\Sigma\colon \Psi;\cdot \xrightarrow{a} a;\Upsilon} \quad \dfrac{}{\Sigma\colon \Psi;\cdot \xrightarrow{b} b;\Upsilon}}{\Sigma\colon \Psi;\cdot \xrightarrow{a \parr b} a,b;\Upsilon}}{\Sigma\colon \Psi;\Delta \xrightarrow{c \parr d \parr e \multimap a \parr b} a,b,\Gamma;\Upsilon}}{\Sigma\colon \Psi;\Delta \longrightarrow a,b,\Gamma;\Upsilon}$$

We can interpret this proof fragment as a reduction of the multiset a,b,Γ to the multiset c,d,e,Γ by backchaining on the clause displayed above.

Of course, a clause may have multiple, top-level implications. In this case, the surrounding context must be manipulated properly to prove the sub-goals that arise in backchaining. Consider a clause of the form

$$G_1 \multimap G_2 \Rightarrow G_3 \multimap G_4 \Rightarrow B_1 \parr B_2$$

labeling the sequent arrow in the sequent $\Sigma\colon \Psi; \Delta \longrightarrow B_1, B_2, \mathcal{A}; \Upsilon$. An attempt to prove this sequent would then lead to attempt to prove the four sequents

$$\Sigma\colon \Psi; \Delta_1 \longrightarrow G_1, \mathcal{A}_1; \Upsilon \qquad \Sigma\colon \Psi; \cdot \longrightarrow G_2; \Upsilon$$

$$\Sigma\colon \Psi; \Delta_2 \longrightarrow G_3, \mathcal{A}_2; \Upsilon \qquad \Sigma\colon \Psi; \cdot \longrightarrow G_4; \Upsilon$$

where Δ is the multiset union of Δ_1 and Δ_2, and $\mathcal{A}$ is the multiset union of $\mathcal{A}_1$ and $\mathcal{A}_2$. In other words, those subgoals immediately to the left of an $\Rightarrow$ are attempted with empty bounded contexts: the bounded contexts, here Δ and $\mathcal{A}$, are divided up to be used to prove those goals immediately to the left of $\multimap$.

2.3. Applications of Forum. Forum specifications have been presented for the operational semantics of programming languages containing side effects, concurrency features, references, exceptions, continuations, and objects [3, 5, 8, 23]. Chirimar [5] used Forum to present the semantics of a RISC processor and Chakravarty [4] used it to specify the logical and operational semantics of a parallel programming language. A specification of a sequent calculus for intuitionistic logic was given by Miller in [23]: that example was improved by Ricci [30], where a proof system for classical logic was also given. The examples in [23, 30] are significantly generalized in this paper.

§3. Representing sequents and inference rules.

Since we now wish to represent one logic and proof system within another, we need to distinguish between the meta-logic, namely, linear logic as presented by Forum, and the various object-logics for which we wish to specify sequent proof systems. Formulas of the object-level will be identified with meta-level terms of type *bool*. Object-level logical connectives will be introduced as needed and as constructors of this type.

A two-sided sequent $\Delta \longrightarrow \Gamma$ is generally restricted so that Δ and Γ are either lists, multisets, or sets of formulas. Sets are used if all three structural rules (exchange, weakening, contraction) are implicit; multisets are used if exchange is implicit; and lists are used if no structural rule is implicit. Since our goal here is to encode object-level sequents into meta-level sequents as directly as possible, and since contexts in Forum are either multisets or sets, we will not be able to represent sequents that make use of lists. It is unlikely, for example, that non-commutative object-logics can be encoded into our linear logic meta theory along the lines we describe below.

3.1. Three schemes for encoding sequents. Consider the well known, two-sided sequent proof systems for classical, intuitionistic, and linear logic. A convenient distinction between these logics can be described, in part, by where the structural rules of thinning and contraction can be applied. In classical logic, these structural rules are allowed on both sides of the sequent arrow; in intuitionistic logic, no structural rules are allowed on the right of the sequent arrow; and in linear logic, they are not allowed on either sides of the arrow. Thus a classical sequent is a pairing of two sets; a linear logic sequent is a pairing of two multisets; and an intuitionistic sequent is the pairing of a set (for the left-hand side) and a multiset (for the right-hand side). This discussion suggests the following representation of sequents in these three 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 $bool \rightarrow o$.

We will identify three schemes for encoding sequents. The *linear scheme* encodes the (object-level) sequent $B_1, \ldots, B_n \longrightarrow C_1, \ldots, C_m$ $(n, m \geq 0)$ by the meta-level formula $\lfloor B_1\rfloor \mathbin{⅋} \cdots \mathbin{⅋} \lfloor B_n\rfloor \mathbin{⅋} \lceil C_1\rceil \mathbin{⅋} \cdots \mathbin{⅋} \lceil C_m\rceil$ or by the Forum sequent

$$\Sigma\colon \cdot; \cdot \longrightarrow \lfloor B_1\rfloor, \ldots, \lfloor B_n\rfloor, \lceil C_1\rceil, \ldots, \lceil C_m\rceil; \cdot.$$

The *intuitionistic scheme* encodes $B_1, \ldots, B_n \longrightarrow C_1, \ldots, C_m$, where $n, m \geq 0$, with the meta-level formula $?\lfloor B_1\rfloor \mathbin{⅋} \cdots \mathbin{⅋} ?\lfloor B_n\rfloor \mathbin{⅋} \lceil C_1\rceil \mathbin{⅋} \cdots \mathbin{⅋} \lceil C_m\rceil$ or by the Forum sequent

$$\Sigma\colon \cdot; \cdot \longrightarrow \lceil C_1\rceil, \ldots, \lceil C_m\rceil; \lfloor B_1\rfloor, \ldots, \lfloor B_n\rfloor.$$

Often intuitionistic sequents are additionally restricted to having one formula on the right. Finally, the *classical scheme* encodes the sequent $B_1, \ldots, B_n \longrightarrow C_1, \ldots, C_m$ $(n, m \geq 0)$ as the meta-level formula

$$?\lfloor B_1\rfloor \mathbin{⅋} \cdots \mathbin{⅋} ?\lfloor B_n\rfloor \mathbin{⅋} ?\lceil C_1\rceil \mathbin{⅋} \cdots \mathbin{⅋} ?\lceil C_m\rceil$$

or by the Forum sequent

$$\Sigma\colon \cdot; \cdot \longrightarrow \cdot; \lfloor B_1\rfloor, \ldots, \lfloor B_n\rfloor, \lceil C_1\rceil, \ldots, \lceil C_m\rceil.$$

The $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ predicates are used to identify which object-level formulas appear on which side of the sequent arrow, and the ? modal is used to mark the formulas to which weakening and contraction can be applied.

3.2. Encoding additive and multiplicative inference rules. We first illustrate how to encode object-level inference rules using the linear scheme.

Consider the specification of the logical inference rules for object-level conjunction, represented here as the infix constant $\wedge$ of type $bool \rightarrow bool \rightarrow bool$.

Consider first the additive inference rules for this connective.

$$\frac{\Delta, A \longrightarrow \Gamma}{\Delta, A \wedge B \longrightarrow \Gamma} \wedge L_1 \quad \frac{\Delta, B \longrightarrow \Gamma}{\Delta, A \wedge B \longrightarrow \Gamma} \wedge L_2 \quad \frac{\Delta \longrightarrow \Gamma, A \quad \Delta \longrightarrow \Gamma, B}{\Delta \longrightarrow \Gamma, A \wedge B} \wedge \mathrm{R}$$

These three inference rules can be specified in Forum using the clauses

$$\begin{array}{llll} (\wedge L_1) & \lfloor A \wedge B \rfloor \circ\!- \lfloor A \rfloor. & (\wedge R) & \lceil A \wedge B \rceil \circ\!- \lceil A \rceil \;\&\; \lceil B \rceil. \\ (\wedge L_2) & \lfloor A \wedge B \rfloor \circ\!- \lfloor B \rfloor. & & \end{array}$$

Let Ψ be a set of formulas that contains the three clauses. The Forum sequent

$$\Sigma\colon \Psi; \cdot \longrightarrow \lfloor B_1 \rfloor, \ldots, \lfloor B_n \rfloor, \lceil C_1 \rceil, \ldots, \lceil C_m \rceil; \cdot$$

can be the conclusion of a decide! rule that selected $(\wedge R)$ rule only if one of the $\lceil \cdot \rceil$-atoms, say $\lceil C_1 \rceil$, is of the form $A \wedge B$ and the sequent

$$\Sigma\colon \Psi; \cdot \longrightarrow \lfloor B_1 \rfloor, \ldots, \lfloor B_n \rfloor, \lceil A \rceil \;\&\; \lceil B \rceil, \lceil C_2 \rceil, \ldots, \lceil C_m \rceil; \cdot$$

is provable. This formula is provable if and only if the two sequents

$$\Sigma\colon \Psi; \cdot \longrightarrow \lfloor B_1 \rfloor, \ldots, \lfloor B_n \rfloor, \lceil A \rceil, \lceil C_2 \rceil, \ldots, \lceil C_m \rceil; \cdot$$

and

$$\Sigma\colon \Psi; \cdot \longrightarrow \lfloor B_1 \rfloor, \ldots, \lfloor B_n \rfloor, \lceil B \rceil, \lceil C_2 \rceil, \ldots, \lceil C_m \rceil; \cdot$$

are provable in Forum. Thus, backchaining on the $(\wedge R)$ clause above can be used to reduce the problem of finding an object-level proof of

$$B_1, \ldots, B_n \longrightarrow A \wedge B, C_2, \ldots, C_m$$

to the problem of finding object-level proofs for

$$B_1, \ldots, B_n \longrightarrow A, C_2, \ldots, C_m \quad \text{and} \quad B_1, \ldots, B_n \longrightarrow B, C_2, \ldots, C_m.$$

Thus, we have successfully captured this right introduction rule for conjunction using *decide*! with the clause corresponding to $(\wedge R)$. A similar and simpler argument shows how left introduction for $\wedge$ is also correctly encoded using the two clauses for $(\wedge L)$. Notice that the two clauses for left introduction could be written equivalently in linear logic as the one formula

$$\lfloor A \wedge B \rfloor \circ\!- \lfloor A \rfloor \oplus \lfloor B \rfloor.$$

(Although $\oplus$ is not a connective of Forum, we shall use it in this fashion in order to write two Forum clauses as one formula.) Thus, these additive rules make use of two (dual) meta-level additive connectives: & and $\oplus$.

Now consider encoding the multiplicative version of conjunction introduction.

$$\frac{\Delta, A, B \longrightarrow \Gamma}{\Delta, A \wedge B \longrightarrow \Gamma} \wedge L \qquad \frac{\Delta_1 \longrightarrow \Gamma_1, A \quad \Delta_2 \longrightarrow \Gamma_2, B}{\Delta_1, \Delta_2 \longrightarrow \Gamma_1, \Gamma_2, A \wedge B} \wedge R$$

It is an easy matter to check that the following two clauses encode these two inference rules.

$$(\wedge L) \quad \lfloor A \wedge B \rfloor \multimapinv \lfloor A \rfloor \parr \lfloor B \rfloor. \qquad (\wedge R) \quad \lceil A \wedge B \rceil \multimapinv \lceil A \rceil \multimapinv \lceil B \rceil.$$

Notice that the clause for right introduction could be written equivalently in linear logic as

$$\lceil A \wedge B \rceil \multimapinv \lceil A \rceil \otimes \lceil B \rceil.$$

Thus, these multiplicative rules make use of two (dual) meta-level multiplicative connectives: $\otimes$ and $\parr$.

Consider now using the classical scheme for representing sequents and consider writing the additive version of the $(\wedge R)$ rule as

$$\lceil A \wedge B \rceil \multimapinv ?\lceil A \rceil \,\&\, ?\lceil B \rceil.$$

In that case, backchaining on this clause would reduce proof search of the sequent

$$\Sigma\colon \Psi; \cdot \longrightarrow \cdot; \lfloor B_1 \rfloor, \ldots, \lfloor B_n \rfloor, \lceil C_1 \rceil, \ldots, \lceil C_m \rceil$$

(where for some i, C_i is $A \wedge B$) to finding proofs for the two sequents

$$\Sigma\colon \Psi; \cdot \longrightarrow ?\lceil A \rceil; \lfloor B_1 \rfloor, \ldots, \lfloor B_n \rfloor, \lceil C_1 \rceil, \ldots, \lceil C_m \rceil$$

and

$$\Sigma\colon \Psi; \cdot \longrightarrow ?\lceil B \rceil; \lfloor B_1 \rfloor, \ldots, \lfloor B_n \rfloor, \lceil C_1 \rceil, \ldots, \lceil C_m \rceil$$

which in turn are provable if and only if the sequents

$$\Sigma\colon \Psi; \cdot \longrightarrow \cdot; \lfloor B_1 \rfloor, \ldots, \lfloor B_n \rfloor, \lceil A \rceil, \lceil C_1 \rceil, \ldots, \lceil C_m \rceil$$

and

$$\Sigma\colon \Psi; \cdot \longrightarrow \cdot; \lfloor B_1 \rfloor, \ldots, \lfloor B_n \rfloor, \lceil B \rceil, \lceil C_1 \rceil, \ldots, \lceil C_m \rceil$$

are provable.

If we had used, instead, the multiplicative encoding of conjunctive introduction,

$$\lceil A \wedge B \rceil \multimapinv ?\lceil A \rceil \otimes ?\lceil B \rceil.$$

a slightly different meta-level proof would have made the same reduction.

It seems natural to consider using a question mark in the head of a clause describing a right or left-introduction rule for classical logic (or just the left-introduction rule for intuitionistic logic). For example, the $(\wedge R)$ rule could have been encoded as

$$?\lceil A \wedge B \rceil \multimapinv ?\lceil A \rceil \,\&\, ?\lceil B \rceil.$$

This encoding style was used in [23, 30], for example. We shall prefer, instead, to encode inference rules without occurrences of question marks in the head of clauses since the structure of meta-level proofs often does not correspond to the structure of object-level proofs. For example, although the sequent

$A \wedge B \wedge C \longrightarrow B$ is provable in classical logic, there is no equivalent object-level proof for the proof displayed below. Here, signatures are not displayed in sequents, $\Gamma = \lceil B \rceil, \lfloor A \wedge B \wedge C \rfloor$, and Ψ is a set of formulas that contains the clause displayed above and *Initial* (see Section 3.4).

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{\dfrac{\dfrac{}{\Psi; \cdot \xrightarrow{\lfloor A\wedge B \rfloor} \cdot; \lfloor A \wedge B \rfloor, \Gamma}}{\Psi; \lfloor A \wedge B \rfloor \longrightarrow \cdot; \lfloor A \wedge B \rfloor, \Gamma}}{\Psi; \lfloor A \wedge B \rfloor \longrightarrow ?\lfloor A \wedge B \rfloor; \Gamma}
\quad
\dfrac{\dfrac{}{\Psi; \cdot \xrightarrow{\lfloor A\wedge B\wedge C \rfloor} \cdot; \Gamma}}{\dfrac{\Psi; \lfloor A \wedge B \wedge C \rfloor \longrightarrow \cdot; \Gamma}{\Psi; \cdot \xrightarrow{?\lfloor A\wedge B\wedge C \rfloor} \cdot; \Gamma}}
}{\Psi; \lfloor A \wedge B \rfloor \xrightarrow{?\lfloor A\wedge B\wedge C \rfloor \mathrel{\circ\!-} ?\lfloor A\wedge B \rfloor} \cdot; \Gamma}
}{\dfrac{\Psi; \lfloor A \wedge B \rfloor \longrightarrow \cdot; \Gamma}{\Psi; \cdot \xrightarrow{?\lfloor A\wedge B \rfloor} \cdot; \Gamma}}
\quad
\dfrac{
\dfrac{
\dfrac{\dfrac{\dfrac{}{\Psi; \cdot \xrightarrow{\lceil B \rceil} \cdot; \lfloor B \rfloor, \Gamma}}{\Psi; \lceil B \rceil \longrightarrow \cdot; \lfloor B \rfloor, \Gamma}}{\Psi; \cdot \xrightarrow{?\lceil B \rceil} \cdot; \lfloor B \rfloor, \Gamma}
\quad
\dfrac{\dfrac{\dfrac{}{\Psi; \cdot \xrightarrow{\lfloor B \rfloor} \cdot; \lfloor B \rfloor, \Gamma}}{\Psi; \lfloor B \rfloor \longrightarrow \cdot; \lfloor B \rfloor, \Gamma}}{\Psi; \cdot \xrightarrow{?\lfloor B \rfloor} \cdot; \lfloor B \rfloor, \Gamma}
}{\Psi; \cdot \xrightarrow{?\lceil B \rceil \parr ?\lfloor B \rfloor} \cdot; \lfloor B \rfloor, \Gamma}
}{\dfrac{\Psi; \cdot \longrightarrow \cdot; \lfloor B \rfloor, \Gamma}{\Psi; \cdot \longrightarrow ?\lfloor B \rfloor; \Gamma}}
}{
\dfrac{\Psi; \cdot \xrightarrow{?\lfloor A\wedge B \rfloor \mathrel{\circ\!-} ?\lfloor B \rfloor} \cdot; \Gamma}{\Psi; \cdot \longrightarrow \cdot; \Gamma}\ \textit{decide!}
}
$$

The fact that "focus" is lost when a question mark is encountered on a formula labeling a sequent arrow means that it is much harder to control the structure of meta-level proofs and to relate them to object-level proofs.

3.3. Encoding quantifier introduction rules. Using the quantification of higher-order types that is available in Forum, it is a simple matter to encode the inference rules for object-level quantifiers. For example, if we use the linear scheme for representing sequents, then the left and right introduction rules for object-level universal quantifier can be written as

$$(\forall L) \quad \lfloor \forall B \rfloor \mathrel{\circ\!-} \lfloor Bx \rfloor. \qquad\qquad (\forall R) \quad \lceil \forall B \rceil \mathrel{\circ\!-} \forall x \lceil Bx \rceil.$$

Here, the symbol $\forall$ is used for both meta-level and object-level quantification: at the object-level $\forall$ has the type $(i \to \mathit{bool}) \to \mathit{bool}$. Thus the variable B above has the type $i \to \mathit{bool}$. Consider the Forum sequent $\Sigma\colon \Psi; \cdot \longrightarrow \lceil \forall B \rceil, \Theta; \cdot$ where Ψ contains the above two clauses. Using *decide*! with the clause for $(\forall R)$ would cause the search for a proof of the above sequent to be reduced to the search for a proof of the sequent $\Sigma, y : i\colon \Psi; \cdot \longrightarrow \lceil By \rceil, \Theta; \cdot$ where y is not present in the signature Σ. Here, the meta-level eigen-variable y also serves the role of an object-level eigen-variable. Dually, consider the Forum sequent $\Sigma\colon \Psi; \cdot \longrightarrow \lfloor \forall B \rfloor, \Theta; \cdot$. Using the *decide*! with the clause for $(\forall L)$ would cause proof search to reduce this sequent to the sequent $\Sigma\colon \Psi; \cdot \longrightarrow \lfloor Bt \rfloor, \Theta; \cdot$ where t is a Σ-term of type i. If we restrict appropriately the use of the type i by constants in Σ, then Σ-terms of type i can be identified with object-level terms.

Notice that the clause for $(\forall L)$ is logically equivalent to the formula

$$\lfloor \forall B \rfloor \mathrel{\circ\!-} \exists x \lfloor Bx \rfloor.$$

Thus, these quantifier rules make use of two (dual) meta-level quantifiers.

3.4. The cut and initial rules. Up to this point, all the Forum clauses used to specify an inference figure have been such that the head of the clause has been an atom. Clauses specifying the cut and initial rules will have heads of rather different structure.

Consider specifying the initial rule (the one asserting that the sequent $B \longrightarrow B$ is provable) using the linear scheme for encoding sequents. The clause

$$(Initial) \qquad \lfloor B \rfloor \parr \lceil B \rceil.$$

will properly encode this rule. Notice that this clause has a head with two atoms (and an empty body). Similarly, the cut rule

$$\frac{\Delta_1 \longrightarrow \Gamma_1, B \quad \Delta_2, B \longrightarrow \Gamma_2}{\Delta_1, \Delta_2 \longrightarrow \Gamma_1, \Gamma_2}\ Cut$$

can be specified simply as the clause

$$(Cut) \qquad \perp \circ\!\!- \lfloor B \rfloor \circ\!\!- \lceil B \rceil.$$

Dually to the initial rule, this clause has an empty head and two bodies.

3.5. Advantages of such encodings. The encoding of an object-level proof system as Forum clauses has certain advantages over encoding them as inference figures. For example, the Forum specifications do not deal with context explicitly and instead they focus on the formulas that are directly involved in the inference rule. The distinction between making the inference rule additive or multiplicative is achieved in inference rule figures by explicitly presenting contexts and either splitting or copying them. The Forum clause representation achieves the same distinction using meta-level additive or multiplicative connectives. Object-level quantifiers can be handled directly using the meta-level quantification. Similarly, the structural rules of contraction and thinning can be captured together using the ? modal. Finally, since the encoding of proof systems is natural and direct, we hope to be able to use the rich meta-theory of linear logic to help in drawing conclusions about object-level proof systems. An example of this kind of meta-level reason will be illustrated in Section 4.4 where a sequent calculus presentation of intuitionistic logic is transformed into a natural deduction presentation by rather simple linear logic equivalences.

Since the encodings of object-level encodings result in logic programs (in the sense of Forum) and since there is significant knowledge and tools available to provide automatic and interactive tools to compute with those logic programs, encodings such as those described here can be important for the automation of various proof systems (see Section 7).

There are, of course, some disadvantages to using linear logic as a meta-theory, the principle one being that it will not be possible to capture all proof systems, such as those for non-commutativity. As we shall see, however, significant and interesting proof systems can be encoded into linear logic and for these systems, broad avenues of meta-level reasoning and automation should be available.

§4. Linear, classical, and intuitionistic logics. In this section, we present Forum encodings of well-known proof systems for linear, classical, and intuitionistic logics. Object-level linear logic will be encoded reusing the same symbols that appear at the meta-level, namely, $!$, $?$, $\otimes$, $⅋$, $\bot$, 1, $\&$, $\oplus$, 0, $\top$ $\multimap$, $\forall_l$, and $\exists_l$. Classical logic is encoded using $\wedge$, $\vee$, $\Rightarrow$, f_c, t_c, $\forall_c$, and $\exists_c$ for conjunction, disjunction, implication, false, true, and universal and existential quantification, respectively, while intuitionistic logic is encoded with $\cap$, $\cup$, $\supset$, f_i, t_i, $\forall_i$, and $\exists_i$ for conjunction, disjunction, implication, false, true, and universal and existential quantification, respectively.

We use the type i to denote object-level individuals and *bool* to denote object-level formulas (our object-logics will all be first-order). All binary connectives have type $bool \to bool \to bool$ and will be written as infix. Object-level constants representing quantification are all of the second order type $(i \to bool) \to bool$: we abbreviated expressions such as $\forall_i(\lambda x.B)$ as simply $\forall_i x B$.

The three signatures Σ_l, Σ_c, and Σ_j will denote the signatures for the object-logics for linear, classical, and intuitionistic, respectively. We assume that each of these signatures also contains the two predicates $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$.

4.1. Three proof systems. Let *LL*, *LK*, and *LJ* denote the be the set of clauses displayed in Figures 2, 3, and 4, respectively.

Proposition 4.1. *The following three correctness statements hold.*

1. *The sequent* $B_1, \ldots, B_n \longrightarrow C_1, \ldots, C_m$ $(m, n \geq 0)$ *has a linear logic proof [13] iff* $\Sigma_l : LL; \cdot \longrightarrow \lfloor B_1 \rfloor, \ldots, \lfloor B_n \rfloor, \lceil C_1 \rceil, \ldots, \lceil C_m \rceil; \cdot$ *has a* $\mathcal{F}$*-proof.*
2. *The sequent* $B_1, \ldots, B_n \longrightarrow C_1, \ldots, C_m$ $(m, n \geq 0)$ *has an LK-proof [12] iff* $\Sigma_c : LK; \cdot \longrightarrow \cdot; \lfloor B_1 \rfloor, \ldots, \lfloor B_n \rfloor, \lceil C_1 \rceil, \ldots, \lceil C_m \rceil$ *has a* $\mathcal{F}$*-proof.*
3. *The sequent* $B_1, \ldots, B_n \longrightarrow B_0$ *has an LJ-proof [12] if and only if the sequent* $\Sigma_j : LJ; \cdot \longrightarrow \lceil B_0 \rceil; \lfloor B_1 \rfloor, \ldots, \lfloor B_n \rfloor$ *has a* $\mathcal{F}$*-proof and the sequent* $B_1, \ldots, B_n \longrightarrow$ *has an LJ-proof iff* $\Sigma_j : LJ; \cdot \longrightarrow \cdot; \lfloor B_1 \rfloor, \ldots, \lfloor B_n \rfloor$ *has a* $\mathcal{F}$*-proof* $(n \geq 0)$.

Proofs are by structural induction of over proof structures. In all cases, proofs in Forum match closely proofs in the corresponding object-logic.

$(\multimap L)$	$\lfloor A \multimap B \rfloor \mathrel{\circ\!-} \lceil A \rceil \mathrel{\circ\!-} \lfloor B \rfloor.$	$(\multimap R)$	$\lceil A \multimap B \rceil \mathrel{\circ\!-} \lfloor A \rfloor \parr \lceil B \rceil.$
$(\otimes L)$	$\lfloor A \otimes B \rfloor \mathrel{\circ\!-} \lfloor A \rfloor \parr \lfloor B \rfloor.$	$(\otimes R)$	$\lceil A \otimes B \rceil \mathrel{\circ\!-} \lceil A \rceil \mathrel{\circ\!-} \lceil B \rceil.$
$(\& L_1)$	$\lfloor A \& B \rfloor \mathrel{\circ\!-} \lfloor A \rfloor.$	$(\& R)$	$\lceil A \& B \rceil \mathrel{\circ\!-} \lceil A \rceil \& \lceil B \rceil.$
$(\& L_2)$	$\lfloor A \& B \rfloor \mathrel{\circ\!-} \lfloor B \rfloor.$	$(\oplus R_1)$	$\lceil A \oplus B \rceil \mathrel{\circ\!-} \lceil A \rceil.$
$(\oplus L)$	$\lfloor A \oplus B \rfloor \mathrel{\circ\!-} \lfloor A \rfloor \& \lfloor B \rfloor.$	$(\oplus R_2)$	$\lceil A \oplus B \rceil \mathrel{\circ\!-} \lceil B \rceil.$
$(\parr L)$	$\lfloor A \parr B \rfloor \mathrel{\circ\!-} \lfloor A \rfloor \mathrel{\circ\!-} \lfloor B \rfloor.$	$(\parr R)$	$\lceil A \parr B \rceil \mathrel{\circ\!-} \lceil A \rceil \parr \lceil B \rceil.$
$(!\, L)$	$\lfloor !\, B \rfloor \mathrel{\circ\!-} ?\lfloor B \rfloor.$	$(!\, R)$	$\lceil !\, B \rceil \Leftarrow \lceil B \rceil.$
$(?\, L)$	$\lfloor ?\, B \rfloor \Leftarrow \lfloor B \rfloor.$	$(?\, R)$	$\lceil ?\, B \rceil \mathrel{\circ\!-} ?\lceil B \rceil.$
$(\forall_l L)$	$\lfloor \forall_l B \rfloor \mathrel{\circ\!-} \lfloor Bx \rfloor.$	$(\forall_l R)$	$\lceil \forall_l B \rceil \mathrel{\circ\!-} \forall x \lceil Bx \rceil.$
$(\exists_l L)$	$\lfloor \exists_l B \rfloor \mathrel{\circ\!-} \forall x \lfloor Bx \rfloor.$	$(\exists_l R)$	$\lceil \exists_l B \rceil \mathrel{\circ\!-} \lceil Bx \rceil.$
$(1L)$	$\lfloor 1 \rfloor \mathrel{\circ\!-} \bot\, .$	$(1R)$	$\lceil 1 \rceil \Leftarrow \top.$
$(\bot\ L)$	$\lfloor \bot \rfloor \Leftarrow \top.$	$(\bot\ R)$	$\lceil \bot \rceil \mathrel{\circ\!-} \bot\, .$
$(0L)$	$\lfloor 0 \rfloor \mathrel{\circ\!-} \top.$	$(\top R)$	$\lceil \top \rceil \mathrel{\circ\!-} \top.$
(Cut)	$\bot \mathrel{\circ\!-} \lfloor B \rfloor \mathrel{\circ\!-} \lceil B \rceil.$	$(Initial)$	$\lfloor B \rfloor \parr \lceil B \rceil.$

FIGURE 2. Forum specification of the *LL* sequent calculus.

$(\Rightarrow L)$	$\lfloor A \Rightarrow B \rfloor \mathrel{\circ\!-} ?\lceil A \rceil \mathrel{\circ\!-} ?\lfloor B \rfloor.$	$(\Rightarrow R)$	$\lceil A \Rightarrow B \rceil \mathrel{\circ\!-} ?\lfloor A \rfloor \parr ?\lceil B \rceil.$
$(\wedge L_1)$	$\lfloor A \wedge B \rfloor \mathrel{\circ\!-} ?\lfloor A \rfloor.$	$(\wedge R)$	$\lceil A \wedge B \rceil \mathrel{\circ\!-} ?\lceil A \rceil \& ?\lceil B \rceil.$
$(\wedge L_2)$	$\lfloor A \wedge B \rfloor \mathrel{\circ\!-} ?\lfloor B \rfloor.$	$(\vee R_1)$	$\lceil A \vee B \rceil \mathrel{\circ\!-} ?\lceil A \rceil.$
$(\vee L)$	$\lfloor A \vee B \rfloor \mathrel{\circ\!-} ?\lfloor A \rfloor \& ?\lfloor B \rfloor.$	$(\vee R_2)$	$\lceil A \vee B \rceil \mathrel{\circ\!-} ?\lceil B \rceil.$
$(\forall_c L)$	$\lfloor \forall_c B \rfloor \mathrel{\circ\!-} ?\lfloor Bx \rfloor.$	$(\forall_c R)$	$\lceil \forall_c B \rceil \mathrel{\circ\!-} \forall x\, ?\lceil Bx \rceil.$
$(\exists_c L)$	$\lfloor \exists_c B \rfloor \mathrel{\circ\!-} \forall x\, ?\lfloor Bx \rfloor.$	$(\exists_c R)$	$\lceil \exists_c B \rceil \mathrel{\circ\!-} ?\lceil Bx \rceil.$
$(f_c L)$	$\lfloor f_c \rfloor \mathrel{\circ\!-} \top.$	$(t_c R)$	$\lceil t_c \rceil \mathrel{\circ\!-} \top.$
(Cut)	$\bot \mathrel{\circ\!-} ?\lfloor B \rfloor \mathrel{\circ\!-} ?\lceil B \rceil.$	$(Initial)$	$\lfloor B \rfloor \parr \lceil B \rceil.$

FIGURE 3. Forum specification of the *LK* sequent calculus.

$(\supset L)$	$\lfloor A \supset B \rfloor \mathrel{\circ\!-} \lceil A \rceil \mathrel{\circ\!-} ?\lfloor B \rfloor.$	$(\supset R)$	$\lceil A \supset B \rceil \mathrel{\circ\!-} ?\lfloor A \rfloor \parr \lceil B \rceil.$
$(\cap L_1)$	$\lfloor A \cap B \rfloor \mathrel{\circ\!-} ?\lfloor A \rfloor.$	$(\cap R)$	$\lceil A \cap B \rceil \mathrel{\circ\!-} \lceil A \rceil \& \lceil B \rceil.$
$(\cap L_2)$	$\lfloor A \cap B \rfloor \mathrel{\circ\!-} ?\lfloor B \rfloor.$	$(\cup R_1)$	$\lceil A \cup B \rceil \mathrel{\circ\!-} \lceil A \rceil.$
$(\cup L)$	$\lfloor A \cup B \rfloor \mathrel{\circ\!-} ?\lfloor A \rfloor \& ?\lfloor B \rfloor.$	$(\cup R_2)$	$\lceil A \cup B \rceil \mathrel{\circ\!-} \lceil B \rceil.$
$(\forall_i L)$	$\lfloor \forall_i B \rfloor \mathrel{\circ\!-} ?\lfloor Bx \rfloor.$	$(\forall_i R)$	$\lceil \forall_i B \rceil \mathrel{\circ\!-} \forall x \lceil Bx \rceil.$
$(\exists_i L)$	$\lfloor \exists_i B \rfloor \mathrel{\circ\!-} \forall x\, ?\lfloor Bx \rfloor.$	$(\exists_i R)$	$\lceil \exists_i B \rceil \mathrel{\circ\!-} \lceil Bx \rceil.$
$(f_i L)$	$\lfloor f_i \rfloor \mathrel{\circ\!-} \top.$	$(t_i R)$	$\lceil t_i \rceil \mathrel{\circ\!-} \top.$
(Cut)	$\bot \mathrel{\circ\!-} ?\lfloor B \rfloor \mathrel{\circ\!-} \lceil B \rceil.$	$(Initial)$	$\lfloor B \rfloor \parr \lceil B \rceil.$

FIGURE 4. Specification of the *LJ* sequent calculus.

$(\Rightarrow L)$	$\lfloor A \Rightarrow B \rfloor \multimapinv \lceil A \rceil \multimapinv \lfloor B \rfloor.$	$(\Rightarrow R)$	$\lceil A \Rightarrow B \rceil \multimapinv \lfloor A \rfloor \parr \lceil B \rceil.$
$(\wedge L_1)$	$\lfloor A \wedge B \rfloor \multimapinv \lfloor A \rfloor.$	$(\wedge R)$	$\lceil A \wedge B \rceil \multimapinv \lceil A \rceil \mathbin{\&} \lceil B \rceil.$
$(\wedge L_2)$	$\lfloor A \wedge B \rfloor \multimapinv \lfloor B \rfloor.$	$(\vee R_1)$	$\lceil A \vee B \rceil \multimapinv \lceil A \rceil.$
$(\vee L)$	$\lfloor A \vee B \rfloor \multimapinv \lfloor A \rfloor \mathbin{\&} \lfloor B \rfloor.$	$(\vee R_2)$	$\lceil A \vee B \rceil \multimapinv \lceil B \rceil.$
$(\forall_c L)$	$\lfloor \forall_c B \rfloor \multimapinv \lfloor Bx \rfloor.$	$(\forall_c R)$	$\lceil \forall_c B \rceil \multimapinv \forall x \lceil Bx \rceil.$
$(\exists_c L)$	$\lfloor \exists_c B \rfloor \multimapinv \forall x \lfloor Bx \rfloor.$	$(\exists_c R)$	$\lceil \exists_c B \rceil \multimapinv \lceil Bx \rceil.$
$(f_c L)$	$\lfloor f_c \rfloor \multimapinv \top.$	$(t_c R)$	$\lceil t_c \rceil \multimapinv \top.$

FIGURE 5. LK_0: The introduction rules of LK with the ? dropped.

$(\supset L)$	$\lfloor A \supset B \rfloor \multimapinv \lceil A \rceil \multimapinv \lfloor B \rfloor.$	$(\supset R)$	$\lceil A \supset B \rceil \multimapinv \lfloor A \rfloor \parr \lceil B \rceil.$
$(\cap L_1)$	$\lfloor A \cap B \rfloor \multimapinv \lfloor A \rfloor.$	$(\cap R)$	$\lceil A \cap B \rceil \multimapinv \lceil A \rceil \mathbin{\&} \lceil B \rceil.$
$(\cap L_2)$	$\lfloor A \cap B \rfloor \multimapinv \lfloor B \rfloor.$	$(\cup R_1)$	$\lceil A \cup B \rceil \multimapinv \lceil A \rceil.$
$(\cup L)$	$\lfloor A \cup B \rfloor \multimapinv \lfloor A \rfloor \mathbin{\&} \lfloor B \rfloor.$	$(\cup R_2)$	$\lceil A \cup B \rceil \multimapinv \lceil B \rceil.$
$(\forall_i L)$	$\lfloor \forall_i B \rfloor \multimapinv \lfloor Bx \rfloor.$	$(\forall_i R)$	$\lceil \forall_i B \rceil \multimapinv \forall x \lceil Bx \rceil.$
$(\exists_i L)$	$\lfloor \exists_i B \rfloor \multimapinv \forall x \lfloor Bx \rfloor.$	$(\exists_i R)$	$\lceil \exists_i B \rceil \multimapinv \lceil Bx \rceil.$
$(f_i L)$	$\lfloor f_i \rfloor \multimapinv \top.$	$(t_i R)$	$\lceil t_i \rceil \multimapinv \top.$

FIGURE 6. LJ_0: The introduction rules of LJ with the ? dropped.

(Pos_1)	$\lceil B \rceil \multimap\, !\lceil B \rceil.$	(Neg_1)	$\lfloor B \rfloor \multimap\, !\lfloor B \rfloor.$
(Pos_2)	$\lfloor B \rfloor \multimapinv\, ?\lfloor B \rfloor.$	(Neg_2)	$\lceil B \rceil \multimapinv\, ?\lceil B \rceil.$
(Cut)	$\perp \multimapinv \lfloor B \rfloor \multimapinv \lceil B \rceil.$	$(Initial)$	$\lfloor B \rfloor \parr \lceil B \rceil.$

FIGURE 7. Some named formulas.

4.2. Modular presentations of classical and intuitionistic logics. The essential difference between the theories LJ and LK is the different set of occurrences of the ? modal. Consider the theories LK_0 and LJ_0 given in Figures 5 and 6. These result from removing the cut and initial rules as well as deleting from the introduction rules of the corresponding LK and LJ theories the ? modal. Define the two new theories

$$LJ' = LJ_0 \cup \{Cut, Initial, Pos_2\} \text{ and } LK' = LK_0 \cup \{Cut, Initial, Pos_2, Neg_2\},$$

where the additional formulas are defined in Figure 7. While LJ' is a strengthening of LJ, they can both prove the same object-level, intuitionistic sequents. Similarly for LK' and LK.

PROPOSITION 4.2. *The following two correctness statements hold.*

1. *Let $B_0, \ldots, B_n$ (for $n \geq 0$) be object-level, intuitionistic formulas. Then*

$$\Sigma_j\colon LJ; \longrightarrow \lceil B_0 \rceil; \lfloor B_1 \rfloor, \ldots, \lfloor B_n \rfloor$$

has a $\mathcal{F}$-proof if and only if $\Sigma_j\colon LJ'; \longrightarrow \lfloor B_1 \rfloor, \ldots, \lfloor B_n \rfloor, \lceil B_0 \rceil;$ has a $\mathcal{F}$-proof.

2. *Let $B_1, \ldots, B_n, C_1, \ldots, C_m$ (for $n, m \geq 0$) be object-level, classical formulas. Then $\Sigma_c\colon LK; \longrightarrow; \lfloor B_1 \rfloor, \ldots, \lfloor B_n \rfloor, \lceil C_1 \rceil, \ldots, \lceil C_m \rceil$ has a $\mathcal{F}$-proof if and only if $\Sigma_c\colon LK'; \longrightarrow \lfloor B_1 \rfloor, \ldots, \lfloor B_n \rfloor, \lceil C_1 \rceil, \ldots, \lceil C_m \rceil;$ has a $\mathcal{F}$-proof.*

Proof We prove the first of these cases since the second is similar.

A consequence of LJ' is the equivalence $\lfloor B \rfloor \equiv ?\lfloor B \rfloor$. Thus we can rewrite the clauses of LJ' into those of LJ by inserting the ? modal. Thus, assuming $\Sigma_j\colon LJ; \longrightarrow \lceil B_0 \rceil; \lfloor B_1 \rfloor, \ldots, \lfloor B_n \rfloor$ has a $\mathcal{F}$-proof, we can use cut-elimination to conclude $\Sigma_j\colon LJ'; \longrightarrow \lceil B_0 \rceil; \lfloor B_1 \rfloor, \ldots, \lfloor B_n \rfloor$ has a $\mathcal{F}$-proof. Using the Pos_2 clauses of LJ' n-times, we can conclude that $\Sigma_j\colon LJ'; \longrightarrow \lfloor B_1 \rfloor, \ldots, \lfloor B_n \rfloor, \lceil B_0 \rceil;$ has a $\mathcal{F}$-proof.

To prove the converse, we prove the following lemma by induction on the height of proofs in Forum: Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be multisets of left-atoms and let R be a right-atom. Then if $\Sigma_j\colon LJ'; \longrightarrow R, \mathcal{L}_1; \mathcal{L}_2$ has a $\mathcal{F}$-proof, then $\Sigma_j\colon LJ; \longrightarrow R; \mathcal{L}_1, \mathcal{L}_2$ has a Forum proof. The proof proceeds by examining each case for how this sequent could be proved. ∎

An immediate corollary of this Proposition and the correctness of LJ and LK is the correctness of LJ' and LK': namely, $\Sigma_j\colon LJ'; \longrightarrow \lfloor B_1 \rfloor, \ldots, \lfloor B_n \rfloor, \lceil B_0 \rceil;$ has a $\mathcal{F}$-proof if and only if the sequent $B_1, \ldots, B_n \longrightarrow B_0$ has an LJ-proof and $\Sigma_c\colon LK'; \longrightarrow \lfloor B_1 \rfloor, \ldots, \lfloor B_n \rfloor, \lceil C_1 \rceil, \ldots, \lceil C_n \rceil;$ has an $\mathcal{F}$-proof if and only if the sequent $B_1, \ldots, B_n \longrightarrow C_1, \ldots, C_n$ has an LK-proof.

Notice that the inference rules for LJ_0 and LK_0 are identical except for a systematic renaming of logical constants. Thus one way to modularly describe the distinction between intuitionistic and classical logics is that the former logic assumes Pos_2 while the latter logic assumes both Pos_2 and Neg_2. This description amounts to saying that contraction is allowed on the right and left in classical proofs but only on the left in intuitionistic proofs.

4.3. Collapsing of modal prefixes. Note that the following equivalences are provable from the various encodings of proof systems:

1. The Cut and Initial rules of LL prove the equivalence $\lceil B \rceil \equiv \lfloor B \rfloor^\perp$.
2. The Cut and Initial rules of LK prove the following equivalences: $\lceil B \rceil \equiv \lfloor B \rfloor^\perp$, $?\lfloor B \rfloor \equiv \lfloor B \rfloor$, $?\lceil B \rceil \equiv \lceil B \rceil$, $?\lceil B \rceil \equiv (?\lfloor B \rfloor)^\perp$, $?\lceil B \rceil \equiv !\lceil B \rceil$, $?\lfloor B \rfloor \equiv !\lfloor B \rfloor$, $!\lfloor B \rfloor \equiv \lfloor B \rfloor$, and $?\lceil B \rceil \equiv \lceil B \rceil$. As an example proof of such an equivalence, the *Cut* rule is equivalent to $(?\lfloor B \rfloor)^\perp \multimapinv ?\lceil B \rceil$. On the other

hand,

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\lfloor B\rfloor \longrightarrow \lfloor B\rfloor}}{\lfloor B\rfloor \longrightarrow ?\lfloor B\rfloor} \quad \dfrac{}{\bot \longrightarrow \bot}}{\lfloor B\rfloor, ?\lfloor B\rfloor \multimap \bot \longrightarrow \bot} \quad \dfrac{\dfrac{}{\lceil B\rceil \longrightarrow \lceil B\rceil}}{\lceil B\rceil \longrightarrow ?\lceil B\rceil}}{\lfloor B\rfloor \parr \lceil B\rceil, ?\lfloor B\rfloor \multimap \bot \longrightarrow ?\lceil B\rceil}$$

That is, the *Initial* rule implies $(?\lfloor B\rfloor)^\bot \multimap ?\lceil B\rceil$. Hence $(?\lfloor B\rfloor)^\bot \equiv ?\lceil B\rceil$ follows from them both.

3. The Cut and Initial rules of *LJ* prove the equivalences $\lceil B\rceil \equiv (?\lfloor B\rfloor)^\bot$, $\lceil B\rceil \equiv !\lceil B\rceil$, $\lfloor B\rfloor \equiv ?\lfloor B\rfloor$, and $\lceil B\rceil \equiv \lfloor B\rfloor^\bot$.

Thus, the cut and initial rules show the (not surprising fact) that $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ are duals of each other. In the cases of *LJ* and *LK*, however, that duality also forces additional equivalences that cause the collapse of some of modals. As it is well known, linear logic has 7 distinct modalities, namely: the empty modality, !, ?, ?!, !?, !?!, and ?!?. Given the *LK* theory, however, all those modals collapse into just two when applied to a $\lfloor\cdot\rfloor$-atom or a $\lceil\cdot\rceil$-atom and in *LJ*, these modals collapse to four when applied to either the $\lceil\cdot\rceil$-atoms or the $\lfloor\cdot\rfloor$-atoms.

Such a collapse is certainly undesirable when specifications rely on proof search: we would like to have a lot of distinctions available to help us understanding how formulas are to be used within object-level proofs. It would be far more interesting to have proof systems for intuitionistic and classical logics, for example, in which these modals would not generally collapse. Recent advances in understanding sequent calculus for these logics provide just such proof systems. We illustrate some of them in Section 5.

4.4. Natural deduction. To illustrate an application of using meta-level reasoning to draw conclusions about an object-logic, we show how a specification for natural deduction in intuitionistic logic can be derived from a sequent calculus specification of intuitionistic logic. For simplicity, we consider a minimal logic fragment of intuitionistic logic involving only $\supset$, $\cap$, and $\forall_i$: let *LM* be the subset of *LJ* from Figure 4 containing *Cut*, *Initial*, and introduction rules for those three connectives. (The disjoint sums are addressed in [23].)

Given the equivalences arising from the cut and initial rules in *LJ* listed in Section 4.3, the specification for $(\supset L)$ is equivalent to the following formulas.

$$\begin{aligned} ?\lfloor B\rfloor \multimap \lceil A\rceil \multimap \lfloor A\supset B\rfloor &\equiv \lfloor B\rfloor \multimap \lceil A\rceil \multimap \lfloor A\supset B\rfloor \\ &\equiv \lceil B\rceil^\bot \multimap \lceil A\rceil \multimap \lceil A\supset B\rceil^\bot \\ &\equiv \lceil A\supset B\rceil \multimap \lceil A\rceil \multimap \lceil B\rceil \end{aligned}$$

The later can be recognized as a specification of the $\supset$ elimination rule. Similarly, the specification for $(\supset R)$ is equivalent to the following formulas.

$$\begin{aligned} ?\lfloor A \rfloor \mathbin{⅋} \lceil B \rceil \multimap \lceil A \supset B \rceil &\equiv \lceil A \rceil^{\perp} \mathbin{⅋} \lceil B \rceil \multimap \lceil A \supset B \rceil \\ &\equiv (!\lceil A \rceil)^{\perp} \mathbin{⅋} \lceil B \rceil \multimap \lceil A \supset B \rceil \\ &\equiv (\lceil A \rceil \Rightarrow \lceil B \rceil) \multimap \lceil A \supset B \rceil \end{aligned}$$

Continuing in such a manner, we can systematically replace all occurrences of $\lfloor \cdot \rfloor$ with occurrences of $\lceil \cdot \rceil$, as listed in Figure 8. The clauses in this figure, named *NM*, can easily be seen as specifying the introduction and elimination rules for this particular fragment of minimal logic. The usual specification of natural deduction rules for minimal logic [11, 16] has intuitionistic implications replacing the top-level linear implications in Figure 8, but as observed in [17], the choice of which implication to use for these top-level occurrences does not change the set of atomic formulas that are provable.

$$\begin{array}{llll} (\supset I) & \lceil A \supset B \rceil \circ\!- \lceil A \rceil \Rightarrow \lceil B \rceil. & (\supset E) & \lceil B \rceil \circ\!- \lceil A \rceil \circ\!- \lceil A \supset B \rceil. \\ (\forall_i I) & \lceil \forall_i B \rceil \circ\!- \forall x \lceil Bx \rceil. & (\forall_i E) & \lceil Bx \rceil \circ\!- \lceil \forall_i B \rceil. \\ (\cap I) & \lceil A \cap B \rceil \circ\!- \lceil A \rceil \mathbin{\&} \lceil B \rceil. & (\cap E_1) & \lceil A \rceil \circ\!- \lceil A \cap B \rceil. \\ & & (\cap E_2) & \lceil B \rceil \circ\!- \lceil A \cap B \rceil. \end{array}$$

FIGURE 8. Specification of the *NM* natural deduction calculus.

As a result of this rather natural connection between clauses in *LM* and *NM*, the following Propositions have rather direct proofs (see [23] for details).

PROPOSITION 4.3. $!\,LM \equiv\, ![(\&NM) \mathbin{\&} \mathit{Initial} \mathbin{\&} \mathit{Cut}]$.

PROPOSITION 4.4. *If B is an object-level formula, then $NM \vdash \lceil B \rceil$ if and only if $LM \vdash \lceil B \rceil$.*

As a consequence of the last Proposition and the correctness of representation of *LM* and *NM*, we can conclude that a formula B has a sequent calculus proof if and only if it has a natural deduction proof.

§5. More refined uses of modals. For the sake of presenting examples in this section, we shall consider the fragments of intuitionistic and classical logics that involve just implication and universal quantification. Gentzen's LJ system for these two connectives is reproduced in Figure 9.

It is well known that proof search in the intuitionistic logic of these connectives can be focused, in the sense that left-introduction rules are only applied to a distinguished formula (such focusing is a justification for backchaining in

$$
\begin{array}{llll}
(\supset L) & \lfloor A \supset B \rfloor \multimap \lceil A \rceil \multimap ?\lfloor B \rfloor. & (\supset R) & \lceil A \supset B \rceil \multimap ?\lfloor A \rfloor \mathbin{\wp} \lceil B \rceil. \\
(\forall_i L) & \lfloor \forall_i B \rfloor \multimap ?\lfloor Bx \rfloor. & (\forall_i R) & \lceil \forall_i B \rceil \multimap \forall x \lceil Bx \rceil. \\
(\mathit{Cut}) & \bot \multimap \lceil A \rceil \multimap ?\lfloor A \rfloor. & (\mathit{Initial}) & \lfloor A \rfloor \mathbin{\wp} \lceil A \rceil.
\end{array}
$$

FIGURE 9. The $\{\supset, \forall_i\}$-fragment of *LJ*

logic programming). Danos et. al. [7] present the focused formulation of intuitionistic logic called *ILU* and displayed in Figure 10. Here, sequents have the form $\Pi; \Gamma \longrightarrow A$ where Γ and Π denote multisets, and Π containing at most one formula. The *ILU* proof system can be encoded in Forum by representing such sequents as $\Sigma\colon \cdot;\cdot \longrightarrow \lfloor \Pi \rfloor, \lceil A \rceil; \lfloor \Gamma \rfloor$ and its inference rules as in Figure 11. (If Γ is a multiset or set of object-level formulas, we write $\lfloor \Gamma \rfloor$ and $\lceil \Gamma \rceil$ to be the corresponding multiset or set of meta-level formulas resulting from applying the corresponding predicate to all formulas in Γ.)

Proofs in ILU are focused in a sense that the left rules $(\supset L)$ and $(\forall_i L)$ can only be applied to formulas in the left linear context Π (in Forum, this is the $\lfloor \cdot \rfloor$-formula without the ?-modal prefix). This restriction, which is enforced using modals in the Forum encoding, constrains proof search significantly.

$$
\frac{}{A; \cdot \longrightarrow A}\ \mathit{initial}
$$

$$
\frac{\Pi; \Gamma_1 \longrightarrow A \quad A; \Gamma_2 \longrightarrow B}{\Pi; \Gamma_1, \Gamma_2 \longrightarrow B}\ \mathit{head\text{-}cut}
\qquad
\frac{;\Gamma_1 \longrightarrow A \quad \Pi; A, \Gamma_2 \longrightarrow B}{\Pi; \Gamma_1, \Gamma_2 \longrightarrow B}\ \mathit{mid\text{-}cut}
$$

$$
\frac{\Pi; \Gamma \longrightarrow A}{\Pi; \Gamma, B \longrightarrow A}\ \mathit{WL}
\qquad
\frac{\Pi; \Gamma, B, B \longrightarrow A}{\Pi; \Gamma, B \longrightarrow A}\ \mathit{CL}
\qquad
\frac{B; \Gamma \longrightarrow A}{\cdot; B, \Gamma \longrightarrow A}\ D
$$

$$
\frac{\cdot; \Gamma \longrightarrow A \quad B; \Gamma' \longrightarrow C}{A \supset B; \Gamma, \Gamma' \longrightarrow C}\ \supset L
\qquad
\frac{\Pi; \Gamma, A \longrightarrow B}{\Pi; \Gamma \longrightarrow A \supset B}\ \supset R
$$

$$
\frac{A[x/t]; \Gamma \longrightarrow B}{\forall_l x A; \Gamma \longrightarrow B}\ \forall_i L
\qquad
\frac{\Pi; \Gamma \longrightarrow A[x/y]}{\Pi; \Gamma \longrightarrow \forall_l x A}\ \forall_l R
$$

FIGURE 10. The sequent calculus *ILU*

$$
\begin{array}{llll}
(\supset L) & \lfloor A \supset B \rfloor \Leftarrow \lceil A \rceil \multimap \lfloor B \rfloor. & (\supset R) & \lceil A \supset B \rceil \multimap ?\lfloor A \rfloor \mathbin{\wp} \lceil B \rceil. \\
(\forall_i L) & \lfloor \forall_i B \rfloor \Leftarrow \lfloor Bx \rfloor. & (\forall_i R) & \lceil \forall_i B \rceil \multimap \forall x \lceil Bx \rceil. \\
(\mathit{Head\text{-}cut}) & \bot \multimap \lceil A \rceil \multimap \lfloor A \rfloor. & (\mathit{Initial}) & \lfloor A \rfloor \mathbin{\wp} \lceil A \rceil. \\
(\mathit{Mid\text{-}cut}) & \bot \Leftarrow \lceil A \rceil \multimap ?\lfloor A \rfloor. & &
\end{array}
$$

FIGURE 11. Specification of the calculus *ILU*

The two cut rules for ILU, *head-cut* and *mid-cut*, are encoded as two formulas in Figure 11 in such as way that the first implies the second: that is,

$$(\lfloor A \rfloor \multimap \lceil A \rceil \multimap \bot) \Rightarrow (?\lfloor A \rfloor \multimap \lceil A \rceil \Rightarrow \bot).$$

is provable in linear logic. As a result, we shall refer to the head-cut as *the* cut rule. Observe that from the *Cut* and *Initial* rules of ILU, we can prove the equivalence $\lceil B \rceil \equiv \lfloor B \rfloor^{\perp}$ but we cannot prove any equivalences between linear logic modals. Note also that *ILU* is equivalent to the neutral fragment of intuitionistic implicational logic of *LU* (see Section 6), although it was formulated in order to obtain a sequent calculus for an *inductive decoration strategy* (see [7] for the definition) of intuitionistic logic into linear logic.

Two sequent calculi, *LKQ* and *LKT*, which provide a focused kind of proof system for classical logic are also presented in [7]. Sequents of the calculus *LKQ* (Figure 12), written as $\Gamma \longrightarrow \Delta; \Pi$ are encoded as Forum sequents $\Sigma\colon \cdot; \cdot \longrightarrow \lceil \Pi \rceil; \lfloor \Gamma \rfloor, \lceil \Delta \rceil$ where Π represents a multiset containing at most one formula. Note that the rules are the same as the ones for the positive classical implicational fragment of *LU* (see Section 6) i.e., rules defined for *positive* formulas. However, *LKQ* cannot be identified with any proper fragment of *LU* since positive polarity is not preserved by the connectives $\Rightarrow$ and $\forall_c$. Sequents of the *LKT* proof system (Figure 13), written as $\Gamma \longrightarrow \Delta; \Pi$, are encoded as $\Sigma\colon \cdot; \cdot \longrightarrow \lfloor \Pi \rfloor; \lfloor \Gamma \rfloor, \lceil \Delta \rceil$, where again Π is a multiset containing at most one formula. Observe that *LKT* is a classical equivalent of *ILU*; that is, the intuitionistic calculus is obtained from *LKT* by the usual restriction of having exactly one formula on the right side of the sequent. *LKT* is equivalent to the negative fragment of classical implicational logic of *LU*.

In both *LKQ* and *LKT* systems there is a collapse of modal prefixes:

1. The *Cut* and *Initial* rules of *LKQ* prove the equivalence $\lceil B \rceil \equiv !\lceil B \rceil$ and the modalities collapse to four when applied to $\lceil \cdot \rceil$-atoms. Thus, in *LKQ* the formula Pos_1 holds.
2. The *Cut* and *Initial* rules of *LKT* prove the equivalence $\lfloor B \rfloor \equiv !\lfloor B \rfloor$ and the modalities collapse to four when applied to $\lfloor \cdot \rfloor$-atoms. Thus, in *LKT* the formula Neg_1 holds.

We present one final example encoding of a proof system, by picking a system that deviates from the previous one in a few details. An optimized version of (the implicational fragment of) *LJ* is presented in Lincoln et. al. [19] (see also [9]). Their system, called *IIL** (Figure 14) does not contain contraction or cut rules, and weakening is only allowed at the leafs of a proof; that is, when the *Initial* rule is applied (to atomic formulas). A key property of *IIL** is that the *principal* formula is not duplicated in the premises of any of the rules. This

$(\Rightarrow L)$	$\lfloor A \Rightarrow B \rfloor \Leftarrow \lceil A \rceil \Leftarrow ?\lfloor B \rfloor.$	$(\Rightarrow R)$	$\lceil A \Rightarrow B \rceil \Leftarrow ?\lfloor A \rfloor \invamp ?\lceil B \rceil.$
$(\forall_c L)$	$\lfloor \forall_c B \rfloor \Leftarrow ?\lfloor Bx \rfloor.$	$(\forall_c R)$	$\lceil \forall_c B \rceil \Leftarrow \forall x\, ?\lceil Bx \rceil.$
(Cut)	$\bot \multimapinv \lceil A \rceil \multimapinv ?\lfloor A \rfloor.$	$(Initial)$	$\lfloor A \rfloor \invamp \lceil A \rceil.$
	$\bot \multimapinv ?\lceil A \rceil \Leftarrow ?\lfloor A \rfloor.$		

FIGURE 12. The calculus *LKQ*

$(\Rightarrow L)$	$\lfloor A \Rightarrow B \rfloor \Leftarrow ?\lceil A \rceil \multimapinv \lfloor B \rfloor.$	$(\Rightarrow R)$	$\lceil A \Rightarrow B \rceil \multimapinv ?\lfloor A \rfloor \invamp ?\lceil B \rceil.$
$(\forall_c L)$	$\lfloor \forall_c B \rfloor \Leftarrow \lfloor Bx \rfloor.$	$(\forall_c R)$	$\lceil \forall_c B \rceil \multimapinv \forall x\, ?\lceil Bx \rceil.$
(Cut)	$\bot \multimapinv ?\lceil A \rceil \multimapinv \lfloor A \rfloor.$	$(Initial)$	$\lfloor A \rfloor \invamp \lceil A \rceil.$
	$\bot \Leftarrow ?\lceil A \rceil \multimapinv ?\lfloor A \rfloor.$		

FIGURE 13. The calculus *LKT*

$(Initial)$	$\lfloor A \rfloor \invamp \lceil A \rceil \multimapinv \top \multimapinv atomic(A).$
$(\supset R)$	$\lceil B \supset C \rceil \multimapinv \lceil B \rceil \invamp \lfloor C \rfloor.$
$(\supset 1L)$	$\lfloor A \supset B \rfloor \invamp \lceil D \rceil \multimapinv \lceil A \rceil \,\&\, (\lfloor B \rfloor \invamp \lceil D \rceil) \multimapinv atomic(A).$
$(\supset 2L)$	$\lfloor (A \supset B) \supset C \rfloor \invamp \lceil D \rceil \multimapinv (\lfloor B \supset C \rfloor \invamp \lceil A \supset B \rceil) \,\&\, (\lfloor C \rfloor \invamp \lceil D \rceil).$

FIGURE 14. The calculus *IIL** where $atomic(\cdot)$ is a predicate of type $bool \to o$ defined to hold for all atomic formulas.

suggests the encoding $\Sigma\colon \cdot;\cdot \longrightarrow \lceil D \rceil, \lfloor \Gamma \rfloor; \cdot$ for the *IIL** sequent $\Gamma \longrightarrow D$. It also requires encoding the *Initial* and $\supset L$ rules differently than we have seen so far: the *Initial* rule uses the additive true, $\top$, to allow weakening, and the $\supset L$ rules uses the additive conjunction, &, to copy the left context and uses a two headed clause so that the right context is not copied but is placed in the correct sequent of the premise.

§6. Using polarities in proof systems. In [14], 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, this corresponds to restricting the use of Pos_2 and Neg_2 rules to *positive* and *negative* formulas respectively and to split the rules for classical, intuitionistic, and linear connectives into cases, depending on the polarities of the subformulas involved.

We proceed to encode LU into Forum as follows. The LU sequent $\Gamma;\Gamma' \longrightarrow \Delta';\Delta$ is encoded as the Forum sequent

$$\Sigma\colon \cdot;\cdot \longrightarrow \lfloor\Gamma\rfloor, \lceil\Delta\rceil; \lfloor\Gamma'\rfloor, \lceil\Delta'\rceil.$$

(Notice the different convention used between LU sequents and Forum sequents with regard to which zones in a sequent allow structural rules.) To encode the polarity of object-level, LU formulas, we introduce three meta-level predicates, $pos(\cdot)$, $neg(\cdot)$, and $neu(\cdot)$, all of type $bool \to o$. We can now encode the Identity and Structure rules on [14, page 206]. The Cut and Initial rules are encoded just as in linear logic. The Cut_2 and Cut_3 rules are, respectively,

$$\bot \multimapinv ?\lceil B\rceil \Leftarrow \lfloor B\rfloor \qquad \text{and} \qquad \bot \Leftarrow \lceil B\rceil \multimapinv ?\lfloor B\rfloor.$$

Both of these formulas, however, are consequences of the Cut rule and are not needed in our encoding of LU. Similarly, the first structural rules are all simple consequences of using exponentials in encoding sequents. Finally, the fact that structural rules are allowed for positive and negative formulas is given as

$$\begin{array}{ll} (Neg) & \lceil N\rceil \multimapinv ?\lceil N\rceil \Leftarrow neg(N). \\ (Pos) & \lfloor P\rfloor \multimapinv ?\lfloor P\rfloor \Leftarrow pos(P). \end{array}$$

Notice that if we use Cut and Initial to eliminate, say, $\lfloor\cdot\rfloor$ for $\lceil\cdot\rceil$ (as we did in Section 4.4), then the only non-trivial inference rules among those coding Identity and Structure rules in this presentation of LU are the (Neg) and (Pos) rules.

The calculus for linear connectives in LU is equivalent to the usual one (see Fig. 2) and the rules do not depend on the polarities. Figure 15 specifies some polarities for classical and intuitionistic connectives (polarities for linear logic connectives can be given similarly). Many of the LU inference rules for classical and intuitionistic connectives are specified in Figure 16. The full encoding of the LU proof system is not given here, but most of it is contained in the union of the clauses in Figures 2, 15, and 16. Observe that the use of Forum to encode the LU proof system provides a reduced set of rules (compare e.g. 8 disjunction rules versus 24 that appear in [14]). Compaction is due partially to the fact that if B is positive then $\lceil B\rceil \equiv\ !\lceil B\rceil$ and $\lfloor B\rfloor \equiv\ ?\lfloor B\rfloor$ and if B is negative then $\lceil B\rceil \equiv\ ?\lceil B\rceil$ and $\lfloor B\rfloor \equiv\ !\lfloor B\rfloor$ (thus the expression $A \Leftarrow \lfloor B\rfloor$ might also be equivalent to $A \multimapinv \lfloor B\rfloor$). Further compaction occurs by departing from Forum syntax slightly and allowing occurrences of $\oplus$ and $!$ in the body of clauses. Such occurrences can be easily removed to form Forum clauses: for example, the formula $(a \oplus !\,b) \multimap c$ is logically equivalent to $(a \multimap c)\ \&\ (b \Rightarrow c)$.

$$
\begin{aligned}
\mathit{notpos}(B) &\Leftarrow \mathit{neg}(B) \oplus \mathit{neu}(B).\\
\mathit{notneg}(B) &\Leftarrow \mathit{pos}(B) \oplus \mathit{neu}(B).\\
\mathit{notneu}(B) &\Leftarrow \mathit{pos}(B) \oplus \mathit{neg}(B).\\
\mathit{pos}(A \wedge B) &\Leftarrow \mathit{pos}(A).\\
\mathit{pos}(A \wedge B) &\Leftarrow \mathit{pos}(B).\\
\mathit{neg}(A \wedge B) &\Leftarrow \mathit{neg}(A) \Leftarrow \mathit{neg}(B).\\
\mathit{neg}(A \vee B) &\Leftarrow \mathit{neg}(A).\\
\mathit{neg}(A \vee B) &\Leftarrow \mathit{neg}(B).\\
\mathit{pos}(A \vee B) &\Leftarrow \mathit{notneg}(A) \Leftarrow \mathit{notneg}(B).\\
\mathit{neg}(A \Rightarrow B) &\Leftarrow \mathit{pos}(A).\\
\mathit{neg}(A \Rightarrow B) &\Leftarrow \mathit{neg}(B).\\
\mathit{pos}(A \Rightarrow B) &\Leftarrow \mathit{notpos}(A) \Leftarrow \mathit{notneg}(B).\\
\mathit{neg}(A \supset B) &\Leftarrow \mathit{neg}(B).\\
\mathit{neg}(\forall_c B)&.\\
\mathit{pos}(\exists_c B)&.\\
\mathit{neu}(A \wedge B) &\Leftarrow \mathit{neu}(A) \Leftarrow \mathit{neu}(B).\\
\mathit{neu}(A \wedge B) &\Leftarrow \mathit{neu}(A) \Leftarrow \mathit{neg}(B).\\
\mathit{neu}(A \wedge B) &\Leftarrow \mathit{neg}(A) \Leftarrow \mathit{neu}(B).\\
\mathit{neu}(A \Rightarrow B) &\Leftarrow \mathit{notneg}(B).
\end{aligned}
$$

FIGURE 15. Positive and negative polarities for intuitionistic and classical logic. See Table 2 in [14].

§7. **Automation of proof systems.** Since the specifications of proof systems are given as clauses in Forum and since Forum can be seen as an *abstract logic programming language* in the sense of [24], it is natural to ask if it is possible to turn these specifications into implementations.

One might attempt to do this using one of the available implementations of Forum [18, 20, 31]. It is, however, a simple matter to turn the specification of Forum given in Figure 1 into a naive interpreter using a logic programming language such as λProlog [25]. We will not present the details of such an implementation except to describe three aspects of it. First, it can be structured such that one inference rule in Figure 1 is translated to one λProlog clause: the resulting implementation is thus rather compact and declarative. Second, the quantification and substitution aspects of the object-logics can be captured directly using λProlog's higher-order features: using a first-order logic programming language such as Prolog would have complicated the implementation significantly. Third, a counter can be used in the clauses of this interpreter to count the number of times a *decide*! or a *decide* rule is used along a particular Forum proof branch.

$$
\begin{array}{ll}
\quad\quad \textit{Identity and structure} & \\
\lfloor B \rfloor \parr \lceil B \rceil. & \\
\quad\quad \bot \multimapinv \lfloor B \rfloor \multimapinv \lceil B \rceil. & \\
\quad\quad \lceil N \rceil \multimapinv ?\lceil N \rceil \Leftarrow neg(N). & \\
\quad\quad \lfloor P \rfloor \multimapinv ?\lfloor P \rfloor \Leftarrow pos(P). & \\
\quad\quad \textit{Conjunction} & \\
\lceil u \wedge v \rceil \Leftarrow \lceil u \rceil \Leftarrow \lceil v \rceil & \Leftarrow pos(u) \oplus pos(v). \\
\lceil u \wedge v \rceil \multimapinv \lceil u \rceil \mathbin{\&} \lceil v \rceil & \Leftarrow notpos(u) \Leftarrow notpos(v). \\
\lfloor u \wedge v \rfloor \multimapinv ?\lfloor u \rfloor \parr ?\lfloor v \rfloor & \Leftarrow pos(u) \oplus pos(v). \\
\lfloor u \wedge v \rfloor \multimapinv \lfloor u \rfloor \oplus \lfloor v \rfloor & \Leftarrow notpos(u) \Leftarrow notpos(v). \\
\quad\quad \textit{Intuitionistic implication} & \\
\lceil u \supset v \rceil \multimapinv ?\lfloor u \rfloor \parr \lceil v \rceil. & \\
\lfloor u \supset v \rfloor \Leftarrow \lceil u \rceil \multimapinv \lfloor v \rfloor. & \\
\quad\quad \textit{Quantifiers} & \\
\lceil \forall_c u \rceil \multimapinv \forall x\, ?\lceil ux \rceil. & \\
\lfloor \forall_c u \rfloor \Leftarrow \lfloor ux \rfloor. & \\
\lceil \exists_c u \rceil \Leftarrow \lceil ux \rceil. & \\
\lfloor \exists_c u \rfloor \multimapinv \forall x\, ?\lfloor ux \rfloor. & \\
\quad\quad \textit{Disjunction} & \\
\lceil u \vee v \rceil \multimapinv !\lceil u \rceil \oplus !\lceil v \rceil & \Leftarrow notneg(u) \Leftarrow notneg(v). \\
\lceil u \vee v \rceil \multimapinv ?\lceil u \rceil \parr ?\lceil v \rceil & \Leftarrow (pos(u) \mathbin{\&} neg(v)) \oplus (neg(u) \mathbin{\&} notneu(v)). \\
\lceil u \vee v \rceil \multimapinv \lceil u \rceil \parr ?!\lceil v \rceil & \Leftarrow neg(u) \Leftarrow neu(v). \\
\lceil u \vee v \rceil \multimapinv ?!\lceil u \rceil \parr \lceil v \rceil & \Leftarrow neu(u) \Leftarrow neg(v). \\
\lfloor u \vee v \rfloor \multimapinv ?\lfloor u \rfloor \mathbin{\&} ?\lfloor v \rfloor & \Leftarrow notneg(u) \Leftarrow notneg(v). \\
\lfloor u \vee v \rfloor \Leftarrow \lfloor u \rfloor \Leftarrow \lfloor v \rfloor & \Leftarrow (pos(u) \mathbin{\&} neg(v)) \oplus (neg(u) \mathbin{\&} notneu(v)). \\
\lfloor u \vee v \rfloor \multimapinv \lfloor u \rfloor \Leftarrow ?\lfloor v \rfloor & \Leftarrow neg(u) \Leftarrow neu(v). \\
\lfloor u \vee v \rfloor \Leftarrow ?\lfloor u \rfloor \multimapinv \lfloor v \rfloor & \Leftarrow neu(u) \Leftarrow neg(v). \\
\quad\quad \textit{Classical implication} & \\
\lceil u \Rightarrow v \rceil \multimapinv ?\lfloor u \rfloor \parr ?\lceil v \rceil & \Leftarrow (neg(u) \mathbin{\&} neg(v)) \oplus (pos(u) \mathbin{\&} notneu(v)). \\
\lceil u \Rightarrow v \rceil \multimapinv \lceil v \rceil \oplus \lfloor u \rfloor & \Leftarrow neg(u) \Leftarrow pos(v). \\
\lfloor u \Rightarrow v \rfloor \multimapinv \lceil u \rceil \mathbin{\&} \lfloor v \rfloor & \Leftarrow neg(u) \Leftarrow pos(v). \\
\lfloor u \Rightarrow v \rfloor \Leftarrow \lceil u \rceil \Leftarrow \lfloor v \rfloor & \Leftarrow (neg(u) \mathbin{\&} neg(v)) \oplus (pos(u) \mathbin{\&} notneu(v)).
\end{array}
$$

FIGURE 16. LU rules

This counter can be used to limit the size of object-level proofs that are searched and in this way, the search for object-level proofs can be controlled in a simple fashion. In general, object-level proofs can be arbitrarily large, so setting a counter such as this is certainly not a complete proof strategy. It is the case, however, that if there is a proof of height h in the object-level, then the interpreter will find a proof if the counter is set to this value. For a number of proofs that we claim below, the value of this counter is often rather small.

To use this prover to prove object-level formulas, one would initialize the prover with the encoding of an object-level proof system and the encoding of the object-level formula. For example, attempting to prove the Forum sequent $LK \Rightarrow \lceil B \rceil$ for some object-level classical logic formula B would correspond to attempting to prove B using the rules of the classical sequent calculus LK. In particular, the single formula intended as LK is the &-conjunction of the universal closure of the clauses listed in Figure 3.

This prover can also be used determine if one collection of inference rules linearly entail other inference rules and equivalences. In particular, all the following can be proved automatically by setting the counter mentioned above to the value 3.

1. The clauses in Figure 3 encoding LK entails the clauses in Figure 4 encoding LJ, at least when these set of clauses are rewritten to use the same set of object-level constants.
2. The clauses in Figure 9 encoding a fragment of LJ entails the clauses in Figure 11 encoding the focused version of LJ called ILU.
3. The forward direction of Proposition 4.3 is easily proved: $\vdash LM \Rightarrow NM$.
4. All the equivalences mentioned in Section 4.3 that arise from the different cut and initial rules used in linear, intuitionistic, and classical logics have simple proofs.

Of course, if such entailments hold, they have immediate consequences for the object-logics that they encode. For example, from the first point above, we know that any (object-level) formula provable in ILU is also provable in LJ.

§8. Conclusion and future work. In this paper, we showed one way that linear logic can be used to specify some sequent calculus proof systems. We presented several examples of such an encoding and argued that such meta-logical encodings can have numerous advantages over the more standard inference figure approach. Since the encodings of the object-level proof systems are natural and direct, the rich meta-theory of linear logic can be used to draw conclusions about object-level proof systems. Because the object-level encodings result in logic programs (in the sense of Forum), the proof systems mentioned in this paper can be easily implemented and some of their properties can be automatically checked.

There is clearly much more to do now that the feasibility of using linear logic in this specification task is clear.

We have not discussed how proof objects can be specified in this setting: adding λ-calculus representations of calculi with natural deduction proofs can probably be done as it is done using an intuitionistic logic meta-theory [10] but such "single-conclusion" proofs would not work in the general sequent calculus setting.

There have been various proposals for non-commutativity variants of classical linear logic [1, 15, 29]: it would be interesting to see if these can be used to capture non-commutative object-level logics in a manner done here.

One reason to use a well understood meta-logic for specification is that it should offer ways to automate many things about inference rules. For example, it seems quite likely that the question whether or not one proof system's encoding linearly entails another proof system's encoding should be decidable, at least in many cases. It is also likely that at least important parts of the proof of cut-elimination for the encoded logic might similarly be automated.

Finally, most interesting proofs that relate provability in two proof systems generally require induction. It seems natural to consider adding to linear logic forms of induction along the lines found in [21, 28].

Acknowledgments. Miller has been funded in part by NSF grants CCR-9803971, INT-9815645, and INT-9815731. Pimentel has been funded by CAPES grant BEX0523/99-2. Pimentel was a visitor at Penn State from September 1999 to January 2001.

REFERENCES

[1] V. Michele Abrusci and Paul Ruet, *Non-commutative logic I: The multiplicative fragment*, ***Annals of Pure and Applied Logic***, vol. 101 (1999), no. 1, pp. 29–64.

[2] Jean-Marc Andreoli, *Logic programming with focusing proofs in linear logic*, ***Journal of Logic and Computation***, vol. 2 (1992), no. 3, pp. 297–347.

[3] Michele Bugliesi, Giorgio Delzanno, Luigi Liquori, and Maurizio Martelli, *A linear logic calculus of objects*, ***Proceedings of the Joint International Conference and Symposium on Logic Programming*** (M. Maher, editor), MIT Press, September 1996.

[4] Manuel M. T. Chakravarty, *On the massively parallel execution of declarative programs*, ***Ph.D. thesis***, Technische Universität Berlin, Fachbereich Informatik, February 1997.

[5] Jawahar Chirimar, *Proof theoretic approach to specification languages*, ***Ph.D. thesis***, University of Pennsylvania, February 1995.

[6] Alonzo Church, *A formulation of the simple theory of types*, ***The Journal of Symbolic Logic***, vol. 5 (1940), pp. 56–68.

[7] Vicent Danos, Jean-Baptiste Joinet, and Harold Schellinx, *LKQ and LKT: sequent calculi for second order logic based upon dual linear decompositions of classical implication*, ***Workshop on linear logic*** (Girard, Lafont, and Regnier, editors), London Mathematical Society Lecture Notes 222, Cambridge University Press, 1995, pp. 211–224.

[8] Giorgio Delzanno and Maurizio Martelli, *Objects in Forum*, ***Proceedings of the International Logic Programming Symposium***, 1995.

[9] Roy Dyckhoff, *Contraction-free sequent calculi for intuitionistic logic*, ***The Journal of Symbolic Logic***, vol. 57 (1992), no. 3, pp. 795–807.

[10] Amy Felty, *A logic program for transforming sequent proofs to natural deduction proofs*, ***Extensions of Logic Programming: International Workshop, Tübingen*** (Peter Schroeder-Heister, editor), LNAI, vol. 475, Springer-Verlag, 1991, pp. 157–178.

[11] Amy Felty and Dale Miller, *Specifying theorem provers in a higher-order logic programming language*, ***Ninth International Conference on Automated Deduction*** (Argonne, IL), Springer-Verlag, May 1988, pp. 61–80.

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

[13] Jean-Yves Girard, *Linear logic*, ***Theoretical Computer Science***, vol. 50 (1987), pp. 1–102.

[14] ———, *On the unity of logic*, ***Annals of Pure and Applied Logic***, vol. 59 (1993), pp. 201–217.

[15] Alessio Guglielmi and Lutz Strassburger, *Non-commutativity and MELL in the calculus of structures*, ***CSL 2001*** (L. Fribourg, editor), LNCS, vol. 2142, 2001, pp. 54–68.

[16] Robert Harper, Furio Honsell, and Gordon Plotkin, *A framework for defining logics*, ***Journal of the ACM***, vol. 40 (1993), no. 1, pp. 143–184.

[17] Joshua Hodas and Dale Miller, *Logic programming in a fragment of intuitionistic linear logic*, ***Information and Computation***, vol. 110 (1994), no. 2, pp. 327–365.

[18] Joshua Hodas, Kevin Watkins, Naoyuki Tamura, and Kyoung-Sun Kang, *Efficient implementation of a linear logic programming language*, ***Proceedings of the 1998 Joint International Conference and Symposium on Logic Programming*** (Joxan Jaffar, editor), 1998, pp. 145 – 159.

[19] Patrick Lincoln, Andre Scedrov, and Natarajan Shankar, *Linearizing intuitionistic implication*, ***Annals of Pure and Applied Logic***, 1993, pp. 151–177.

[20] P. López and E. Pimentel, *The UMA Forum linear logic programming language*, implementation, January 1998.

[21] Raymond McDowell and Dale Miller, *Cut-elimination for a logic with definitions and induction*, ***Theoretical Computer Science***, vol. 232 (2000), pp. 91–119.

[22] Dale Miller, *The π-calculus as a theory in linear logic: Preliminary results*, ***Proceedings of the 1992 Workshop on Extensions to Logic Programming*** (E. Lamma and P. Mello, editors), LNCS, no. 660, Springer-Verlag, 1993, pp. 242–265.

[23] ———, *Forum: A multiple-conclusion specification language*, ***Theoretical Computer Science***, vol. 165 (1996), no. 1, pp. 201–232.

[24] Dale Miller, Gopalan Nadathur, Frank Pfenning, and Andre Scedrov, *Uniform proofs as a foundation for logic programming*, ***Annals of Pure and Applied Logic***, vol. 51 (1991), pp. 125–157.

[25] Gopalan Nadathur and Dale Miller, *An Overview of λProlog*, ***Fifth International Logic Programming Conference*** (Seattle), MIT Press, August 1988, pp. 810–827.

[26] Lawrence C. Paulson, *The foundation of a generic theorem prover*, ***Journal of Automated Reasoning***, vol. 5 (1989), pp. 363–397.

[27] Frank Pfenning, *Elf: A language for logic definition and verified metaprogramming*, ***Fourth Annual Symposium on Logic in Computer Science*** (Monterey, CA), June 1989,

pp. 313–321.

[28] Elaine Gouvêa Pimentel, *Lógica linear e a especificação de sistemas computacionais*, ***Ph.D. thesis***, Universidade Federal de Minas Gerais, Belo Horizonte, M.G., Brasil, December 2001 (written in English).

[29] Christian Retoré, *Pomset logic: a non-commutative extension of classical linear logic*, ***Proceedings of TLCA***, vol. 1210, 1997, pp. 300–318.

[30] Giorgia Ricci, *On the expressive powers of a logic programming presentation of linear logic (FORUM)*, ***Ph.D. thesis***, Department of Mathematics, Siena University, December 1998.

[31] Christian Urban, *Forum and its implementations*, ***Master's thesis***, University of St. Andrews, December 1997.

E-mail: dale@cse.psu.edu

COMPUTER SCIENCE AND ENGINEERING, 220 POND LAB,
PENNSYLVANIA STATE UNIVERSITY, UNIVERSITY PARK, PA 16802-6106 USA

E-mail: pimentel@dcc.ufmg.br

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO,
UNIVERSIDADE FEDERAL DE MINAS GERAIS, BELO HORIZONTE, M.G. BRASIL