

---

# Some 3-Dimensional Systems of Labelled Deduction

DICK OEHRLE, *Department of Linguistics, University of Arizona, Tucson, AZ 85721 USA. E-mail: oehrle@ccit.arizona.edu*

One way to individuate logical systems is according to the consequence relation they define. But there are distinctions of finer grain. Gabbay [11] suggests that ‘a logical system  $\mathbf{L}$  is not just the traditional consequence relation  $\vdash$  (monotonic or non monotonic) but a pair  $(\vdash, S_{\vdash})$ , where  $\vdash$  is a mathematically defined consequence relation (ie the set of pairs  $(\Delta, Q)$  such that  $\Delta \vdash Q$ ) satisfying whatever minimal conditions on a consequence one happens to agree to, and  $S_{\vdash}$  is an algorithmic system for generating all those pairs’ (p. 10). And Lambek [20] writes:

‘When studying systems such as the syntactic calculus, one may take one of several points of view, regarding the system as

- (a) a deductive system (no equality relation between arrows is considered);
- (b) a poset (all arrows from  $A$  to  $B$  are taken to be the same and one usually writes  $A \leq B$  for  $A \rightarrow B$  and  $A = B$  for  $A \leftrightarrow B$ );
- (c) a pre-ordered set (as in (b), except that  $A$  and  $B$  are not necessarily regarded as equal when  $A \leftrightarrow B$ );
- (d) a category (arrows are labelled and an equality relation between arrows is carefully chosen to ensure, for example, that  $\otimes$  is a bifunctor and that  $- \otimes B$  is left adjoint to  $-/B$ ).

There are parallel developments in linguistics and the study of formal languages. Linguistic systems are multi-dimensional [28], involving associations among phonological structure, semantic and pragmatic representation, and syntactic type. The perspective of *labelled deductive systems* [11] provides a natural setting in which to investigate the thesis that the phonological and interpretive aspects of linguistic representations may be regarded as algebraic structures, and to model the analysis of utterances by labelling syntactic categories with terms of these algebras. From this point of view, it is possible to investigate the coordination of information across dimensions, and to see how properties of the labelling algebras contribute to the deductive power of various systems. Within this setting, interest shifts from the question of generative capacity to the study of the relations among terms and types characterized by a labelled consequence relation.

We begin (§1) by considering properties of the Curry–Howard correspondence between proofs and  $\lambda$ -terms in the setting of a sequent presentation of Lambek’s associative syntactic calculus ([18]) and some of its neighbors. As is well known, in this setting, the correspondence is not an isomorphism. But it does define an equivalence relation on proofs: equivalent proofs have the same endsequent labelled with

the same  $\lambda$ -term. A question of interest, then, is whether, for any sequent  $\Sigma$ , it is possible to determine *on the basis of its form*, which equivalence classes of proofs it constitutes the endsequent of. We examine this problem in §2, and our examination leads to the idea (developing insights of [24]) that the constraints imposed by proof structure on the matching of atomic resources can be expressed as well by an appropriate system of labelling. In §3, we present, in a schematic way, a number of systems of labelled deduction along the desired lines and show the equivalence of a variety of different presentations. In contrast to other systems of resource-labelling in the categorial literature ([24, 27, 8]), the systems studied here are based on a type language which does not distinguish left- and right-residuals: all information about relative order and connectedness is expressed in the term system. To demonstrate the linguistic interest of this family of systems, we show in §4 how a particular system of this kind—one with no corresponding simple relative among the neighbors of the Lambek calculus—provides an elegant model of quantificational type structure and quantifier scope ambiguities. Finally, in §5, we consider the range of options allowed by making different kinds of choices concerning the structure of the terms of the resource labelling system, and how these choices are related to the more familiar topography of the categorial landscape.

## 1 Sequent calculus with $\lambda$ -terms

Sequent characterizations of deducibility relations reveal directly a spectrum of different substructural logics. Of special interest here is the distinction in sequent presentations between structural rules and logical rules. Logical rules characterize the deductive behavior of the logical connectives. Structural rules characterize the resource-sensitivity of sequents. The sequent presentation also has other proof-theoretical advantages, of course—advantages that stem from Cut elimination and the subformula property. But it has disadvantages as well: notational bookkeeping of inactive structures and permutability of inference steps. (Wallen [33] offers a very clear exposition of these issues.) We will encounter some of these disadvantages below in discussing the form the Curry–Howard correspondence takes in the sequent setting.

In the postulates to be presented below, each sequent is associated with a  $\lambda$ -term according to the Curry–Howard correspondence [16]. In fact, we will label types with typed  $\lambda$ -terms, and the postulates will define a system of labelled deduction. To state the postulates, we need some preliminary definitions.

*Types.* Let  $\mathcal{A}$  be a nonempty set of *atomic types* (or: syntactic atoms). The set  $\mathcal{F}$  of formulae is defined by closing this set under three binary operations  $\bullet$ ,  $/$ , and  $\backslash$ :

$$\mathcal{F} ::= \mathcal{A} \mid \mathcal{F} \bullet \mathcal{F} \mid \mathcal{F} / \mathcal{F} \mid \mathcal{F} \backslash \mathcal{F}$$

*Terms.* By a  $\lambda$ -type symbol, we mean either: (1) an atomic  $\lambda$ -type, or given two  $\lambda$ -type symbols  $\alpha$  and  $\beta$ , (2) the product type  $\langle \alpha, \beta \rangle$  or (3) the implicational type  $\alpha \rightarrow \beta$ . Terms have the structure  $\tau^\alpha$ , where  $\alpha$  is a type symbol, and such a term is said to be a term of type  $\alpha$ . For each type symbol  $\alpha$ , we have variables  $x_1^\alpha, x_2^\alpha, \dots$  (with sub- and super-scripts suppressed whenever readability allows). In addition, we have terms formed by *abstraction* and *application*, and by *pairing* and *projection*:

- $\lambda$ -abstraction: given term  $\varphi^\beta$  and variable  $x^\alpha$ , form  $(\lambda x^\alpha.\varphi^\beta)^{\alpha \rightarrow \beta}$ ;  
 application: given  $\psi^{\alpha \rightarrow \beta}$  and  $\xi^\alpha$ , form  $(\psi^{\alpha \rightarrow \beta} \xi^\alpha)^\beta$ ;  
 pairing: given terms  $\varphi^\alpha$  and  $\psi^\beta$ , form  $\langle \varphi^\alpha, \psi^\beta \rangle^{\alpha, \beta}$ ;  
 projection: given term  $\chi^{\alpha, \beta}$ , form terms  $(\pi_1 \chi^{\alpha, \beta})^\alpha$  and  $(\pi_2 \chi^{\alpha, \beta})^\beta$

These operations are assumed to conform to the standard equations [14]:

$$\pi_1 \langle \alpha, \beta \rangle = \alpha, \quad \pi_2 \langle \alpha, \beta \rangle = \beta, \quad \langle \pi_1 \chi, \pi_2 \chi \rangle = \chi$$

$$((\lambda x^\alpha.\varphi^\beta)^{\alpha \rightarrow \beta} u^\alpha) = [u/x] \varphi^\beta, \quad (\lambda x^\alpha.(t^{\alpha \rightarrow \beta} x^\alpha))^{\alpha \rightarrow \beta} = t^{\alpha \rightarrow \beta}$$

And substitution and  $\beta$ -reduction are defined in the standard way [15]. We write  $t \triangleright t'$  when  $t'$  is the result of replacing one or more  $\beta$ -redexes (that is, subterms of the form  $((\lambda x^a \varphi^b) v^a)$ ) in  $t$  by their contracted forms (corresponding subterms of the form  $[v^a/x^a] \varphi^b$ ) or by replacing one or more occurrences of projection operators (of the form  $\pi_1 \langle u, v \rangle$  or  $\pi_2 \langle u, v \rangle$ ) by their projections (of the forms  $u$  and  $v$ , respectively). If  $t$  is a term for which there is no  $t'$  with  $t \triangleright t'$ , we call  $t$  *normal*.

*Pure  $\lambda$ -terms* are built up entirely from variables, by abstractions, applications, pairings and projections. We will consider below applications in which it is useful to build terms up from constants of certain types as well (as, for example, in the system of intensional logic of Montague's *PTQ* [21]).

*Type constraints.* Let *typ* be a function with domain the set  $\mathcal{A}$  of syntactic atoms and codomain the set of  $\lambda$ -type symbols. We extend *typ* to a function from  $\mathcal{F}$  to the set of  $\lambda$ -type symbols in the usual way, by the additional clauses:

$$\begin{aligned} \text{typ}(A \bullet B) &= \langle \text{typ}(A), \text{typ}(B) \rangle \\ \text{typ}(A/B) &= \text{typ}(B \setminus A) = \text{typ}(B) \rightarrow \text{typ}(A) \end{aligned}$$

*Labelled Types.* A labelled type is a pair written  $\tau^\alpha : T$ , with  $\tau^\alpha$  a term of  $\lambda$ -type  $\alpha$  and  $\alpha = \text{typ}(T)$ . Superscripts will be generally suppressed below.

*Sequents.* An (intuitionistic) sequent is a pair  $\langle \Gamma, u : A \rangle$ , displayed  $\Gamma \vdash u : A$ , where  $u : A$  is a labelled type with  $u$  a term in normal form and  $\Gamma$  is a *structure*—that is, an element of the set  $\mathcal{S}$  defined below:

$$\mathcal{S} ::= \mathcal{F} \mid (\mathcal{S}, \mathcal{S})$$

In the sequent  $\Gamma \vdash A$ , we call  $\Gamma$  the *antecedent* and  $A$  the *succedent*. As here, we shall use  $A, B, C, \dots$  as variables ranging over types,  $t, u, v, w, \dots$  as variables ranging over terms, and  $\Gamma, \Delta, \Theta, \dots$  as variables ranging over structures.  $\Gamma[\Delta]$  denotes a formula containing a designated occurrence of the subformula  $\Delta$ ; when  $\Gamma[\Delta]$  occurs in the premise of a rule and  $\Gamma[\Delta']$  occurs in the conclusion, it is to be understood that  $\Delta'$  is to be substituted for the designated occurrence of  $\Delta$  in premise occurrence of  $\Gamma[\Delta]$ .

*Postulates.* The sequent calculi of interest here are defined by the following axiom and inference rules. All the systems enjoy the Identity Axiom, Cut, and all the logical rules governing the deductive behavior of the type-constructors  $\bullet$ ,  $\setminus$ , and  $/$  in antecedent and succedent.

**Identity Axiom**

$$u : A \vdash u' : A \quad u \triangleright u'$$

**Logical Rules**

$$L/ \frac{\Delta \vdash u : B \quad \Gamma[tu : A] \vdash C}{\Gamma[(t : A/B, \Delta)] \vdash C} \quad R/ \frac{(\Gamma, u : A) \vdash t : B}{\Gamma \vdash \lambda u.t : B/A} \quad u \text{ not free in } \Gamma$$

$$L\backslash \frac{\Delta \vdash u : B \quad \Gamma[tu : A] \vdash C}{\Gamma[(\Delta, t : B\backslash A)] \vdash C} \quad R\backslash \frac{(u : A, \Gamma) \vdash t : B}{\Gamma \vdash \lambda u.t : A\backslash B} \quad u \text{ not free in } \Gamma$$

$$L\bullet \frac{\Gamma[(\pi_1 u : A, \pi_2 u : B)] \vdash C}{\Gamma[u : A \bullet B] \vdash C} \quad R\bullet \frac{\Gamma \vdash u : A \quad \Delta \vdash v : B}{(\Gamma, \Delta) \vdash \langle u, v \rangle : A \bullet B}$$

**Structural Rules**

$$\text{Cut} \frac{\Gamma \vdash u : A \quad \Delta[t : A] \vdash v : B}{\Delta[\Gamma] \vdash v : B} \quad t \triangleright u$$

$$\text{Permutation} \frac{\Gamma[(u : A, v : B)] \vdash w : C}{\Gamma[(v : B, u : A)] \vdash w : C}$$

$$\text{Associativity } 1 \frac{\Gamma[(u : A, (v : B, w : C))] \vdash t : D}{\Gamma[((u : A, v : B), w : C)] \vdash t : D}$$

$$\text{Associativity } 2 \frac{\Gamma[((u : A, v : B), w : C)] \vdash t : D}{\Gamma[(u : A, (v : B, w : C))] \vdash t : D}$$

The system whose only structural rule is Cut—also called the ‘non-associative syntactic calculus’ [19]—we call **NL**. Adding the associativity rules gives Lambek’s syntactic calculus **L** [18].<sup>1</sup> The addition of Permutation to **L** yields the Lambek/van Benthem calculus **LP**. Adding Permutation to **NL** yields the permutation-closed variant of **NL** called **NLP**. Cut-elimination holds for all these systems.

In the sections that follow, the fact that each atomic subformula of the endsequent of a proof can be traced back to a unique atomic formula in an axiom-leaf, in the obvious way defined by the postulates, will be useful.

---

<sup>1</sup>We won’t consider here the systems resulting from the addition of just one or the other of the associativity rules, although Hoeksema’s observation that there is no evidence for a ‘Left Node Raising’ construction symmetric to the well-established ‘Right Node Raising’ construction suggests that the two associativity rules need not be coupled. For discussion, see [26].

1.1 *The correspondence of proofs and terms*

The theoretical interest of the Curry–Howard morphism lies in the isomorphism it affords between natural deduction proofs for the intuitionistic logic of conjunction and implication and the set of pure  $\lambda$ -terms. With regard to the systems discussed above, the purity of the Curry–Howard morphism is marred by two kinds of imperfection. First, there may not be—and in general, in the systems defined here, are not—enough proofs for all the different kinds of  $\lambda$ -terms available. (In other words, given our definition of  $\lambda$ -term, the morphism as defined is not surjective.) Second, different proofs may be associated by the correspondence with the same  $\lambda$ -term. (Meaning that the morphism as defined is not injective.) This is already obvious from the similarity of terms in the rules for  $/$  and  $\backslash$ , but also arises from permutability of antecedent inference steps. Nonsurjectivity raises the possibility of characterizing restricted sets of  $\lambda$ -terms with the aim of finding matches between sets of terms and proofs (in one or another deductive system): [7, 3, 12]. Noninjectivity raises the question of whether there are proofs systems for which the correspondence between proofs and terms is 1-1.

An example of equivalent proofs in  $\mathbf{L}$  is presented below (where associativity inferences are suppressed and antecedent formulas are treated simply as sequences of labelled types):

$$\frac{\frac{g(x) : B \vdash g(x) : B \quad f(g(x)) : A \vdash f(g(x)) : A}{x : C \vdash x : C \quad f : A/B, g(x) : B \vdash f(g(x)) : A}}{f : A/B, g : B/C, x : C \vdash f(g(x)) : A}}{f : A/B, g : B/C \vdash \lambda x.f(g(x)) : A/C}$$

$$\frac{\frac{x : C \vdash x : C \quad g(x) : B \vdash g(x) : B}{g : B/C, x : C \vdash g(x) : B} \quad f(g(x)) : A \vdash f(g(x)) : A}{f : A/B, g : B/C, x : C \vdash f(g(x)) : A}}{f : A/B, g : B/C \vdash \lambda x.f(g(x)) : A/C}$$

The equivalence displayed by the pair of proofs above should be contrasted with the pair of nonequivalent proofs below:

$$\frac{\frac{f(p) : S \vdash f(p) : S \quad g(f(p)) : S \vdash g(f(p)) : S}{p : S \vdash p : S \quad f(p) : S, g : S \backslash S \vdash g(f(p)) : S}}{f : S/S, p : S, g : S \backslash S \vdash g(f(p)) : S}$$

$$\frac{\frac{g(p) : S \vdash g(p) : S \quad f(g(p)) : S \vdash f(g(p)) : S}{p : S \vdash p : S \quad f : S/S, g(p) : S \vdash f(g(p)) : S}}{f : S/S, p : S, g : S \backslash S \vdash f(g(p)) : S}$$

Comparison of the two pairs of proofs shows that in the first pair, the atomic subformulae of the endsequent that are paired in axiom leaves are the same in the equivalent proofs, whereas this property fails to hold of the second pair. This suggests that we examine how identity of axiom-leaf bindings bears on equivalence with respect to Curry–Howard properties. Such an examination leads to the following

## THEOREM 1.1

Any two cut-free proofs of the same labelled endsequent  $\Gamma \vdash u : A$  have the same set of axiom-leaf pairs and the same correspondence between atomic subformulae of the endsequent and axiom-leaf occurrences.

PROOF. The theorem follows by decorating the types of the endsequent with a polarity (1 for antecedent types; 0 for the succedent type) and pushing both polarities and terms onto the subformulae of complex types by recursively applying the following recipe:

$$\begin{array}{ll}
L\bullet & \langle x, y \rangle : (A \bullet B)^1 \rightsquigarrow \langle x, y \rangle : (x : A^1 \bullet y : B^1)^1 \\
R\bullet & \langle x, y \rangle : (A \bullet B)^0 \rightsquigarrow \langle x, y \rangle : (x : A^0 \bullet y : B^0)^0 \\
L/ & t : (A/B)^1 \rightsquigarrow t : (tu : A^1 / u : B^0)^1 \\
L\backslash & t : (B\backslash A)^1 \rightsquigarrow t : (u : B^0 \backslash A : tu)^1 \\
R/ & \lambda x. \varphi : (A/B)^0 \rightsquigarrow \lambda x. \varphi : (\varphi : A^0 / x : B^1)^0 \\
R\backslash & \lambda x. \varphi : (B\backslash A)^0 \rightsquigarrow \lambda x. \varphi : (x : B^1 \backslash \varphi : A^0)^0
\end{array}$$

These steps are deterministic: for the cases  $L\bullet, R\bullet, R/, R\backslash$ , the term associated with the complex type carries all the information needed to assign terms to the immediated subformulae; for the cases  $L/$  and  $L\backslash$ , the proof as a whole provides the appropriate information. (Note that in these rules  $u$  must be chosen identically in both proofs: if we choose  $u$  in one proof and a distinct term  $u'$  in the other, then one of the terms of the endsequent will contain different subterms— $tu$  in one proof and  $tu'$  in other—contrary to the hypothesis that the endsequents are the same.) Moreover, in any proof, viewed from the bottom up, the logical rules—in whatever order they apply—perform exactly the same decomposition. The pairing of the labelled atoms that results is uniquely determined, by the fact that pairs contain exactly the same free atomic subterms. ■

A counterpart to this result is the following:

## THEOREM 1.2

If two proofs whose endsequents differ at most in their term-labelling have the same axiom-leaf bindings, then the labelling of the endsequents is the same.

PROOF. Assign corresponding term variables to the types of the endsequent, choosing fresh variables (the same in each proof) at each step. The terms of the axiom-leaves—the same in each proof—act as ground terms for the terms of the atomic subformulas. We use this information systematically to assign terms to complex types—an assignment which at each stage is determined completely by the terms associated with the component subtypes. ■

These results suggest that we attempt to determine directly what axiom-bindings are allowed by valid proofs. We will investigate this first for the system of unlabelled types corresponding to **L**.

## 2 Axiom-leaf pairings and proof structure

Gentzen [13] expressed the idea that the form of a valid formula encodes its proof. In the *occurrence logics* considered here—logics lacking the structural rules Thinning

(or Weakening or Monotonicity), Contraction, and Expansion—this idea is especially perspicuous, because each atomic subformula of the endsequent of a proof can be traced back to a single occurrence in an axiom-leaf pairing. Since each axiom pairs two atomic types, we have the following results (related to van Benthem’s ‘count’ invariant):

**THEOREM 2.1**

The number of occurrences of any given atomic type among the subformulas of a valid sequent in an occurrence logic is even.

**COROLLARY 2.2**

The number of atomic subformulas of any valid sequent in an occurrence logic is even.

By the corollary, one can tell by counting atomic subformulas that such sequents as  $A \vdash B \setminus A$  (the counterpart of the intuitionistic tautology  $A \rightarrow (B \rightarrow A)$ ) and  $A \bullet A \vdash A$  are not valid in occurrence logics.

Now, any proof of a sequent pairs up atomic subformulas in axiom leaves. But not any pairing of axiom leaves of the atomic subformulas of a valid sequent is associated with a proof of it. As an example, consider again the cut-free proofs of the sequent  $S/S, S, S \setminus S \vdash S$ , displayed below with the atomic subformulas individuated with subscripts:

$$\frac{\frac{S_3 \vdash S_2 \quad \frac{S_1 \vdash S_4 \quad S_5 \vdash S_6}{S_1, S_4 \setminus S_5 \vdash S_6}}{S_1/S_2, S_3, S_4 \setminus S_5 \vdash S_6} \quad \frac{S_5 \vdash S_2 \quad S_1 \vdash S_6}{S_1/S_2, S_5 \vdash S_6}}{S_1/S_2, S_3, S_4 \setminus S_5 \vdash S_6} \quad \frac{S_3 \vdash S_4 \quad S_5 \vdash S_2}{S_3, S_4 \setminus S_5 \vdash S_2} \quad \frac{S_1 \vdash S_6}{S_1 \vdash S_6}}{S_1/S_2, S_3, S_4 \setminus S_5 \vdash S_6} \quad \frac{S_3 \vdash S_2 \quad S_1 \vdash S_4}{S_1/S_2, S_3 \vdash S_4} \quad \frac{S_5 \vdash S_6}{S_5 \vdash S_6}}{S_1/S_2, S_3, S_4 \setminus S_5 \vdash S_6}$$

These proofs exhaust the Cut-free proof-space. Since the atomic subformulas of the the endsequent consist of 6 occurrences of the atomic type  $S$ , there are 15 possible ways of pairing up these occurrences. Only 2 of these 15 correspond to proofs of the endsequent. An interesting property of the axiom leaves of valid proofs, however, is that each atomic subformula  $S_i$  of the endsequent ends up either as the antecedent formula of an axiom leaf in all proofs or as the consequent formula of an axiom leaf in all proofs. This is not accidental: it is an invariant of the proof space.

Indirectly, we have already seen why: label each type with one of two polarities, as above: antecedent types with 1, succedent types with 0. (Thus, axiom instances are labelled  $A^1 \vdash A^0$ .) Examination of the logic rules then justifies the recursive steps:

$$\begin{aligned} (L\bullet) \quad (A \bullet B)^1 &\rightsquigarrow (A^1 \bullet B^1) \\ (R\bullet) \quad (A \bullet B)^0 &\rightsquigarrow (A^0 \bullet B^0) \\ (L/) \quad (A/B)^1 &\rightsquigarrow (A^1/B^0) \\ (L\setminus) \quad (B \setminus A)^1 &\rightsquigarrow (B^0 \setminus A^1) \\ (R/) \quad (A/B)^0 &\rightsquigarrow (A^0/B^1) \\ (R\setminus) \quad (B \setminus A)^0 &\rightsquigarrow (B^1 \setminus A^0) \end{aligned}$$

Eventually, these rules lead to the assignment of polarities to all atomic subformulas of a sequent, as illustrated below:

$$S_1/S_2, S_3, S_4 \setminus S_5 \vdash S_6 \rightsquigarrow S_1^1/S_2^0, S_3^1, S_4^0 \setminus S_5^1 \vdash S_6^0$$

Now, instead of considering 15 possible pairings of atomic subformulas, we may consider only those pairings which involve occurrences of the same atomic type of opposite polarity. The atomic polar decomposition represented by the multiset  $[S_1^1, S_2^0, S_3^1, S_4^0, S_5^1, S_6^0]$  of signed atomic types allows 6 pairings of signed atoms of the same type but opposite polarity, displayed below. As we know, only two of these are associated with proofs of the sequent in question.

			proof bindings?
$S_1^1, S_2^0$	$S_3^1, S_4^0$	$S_5^1, S_6^0$	no
$S_1^1, S_2^0$	$S_3^1, S_6^0$	$S_4^0, S_5^1$	no
$S_1^1, S_4^0$	$S_2^0, S_3^1$	$S_5^1, S_6^0$	yes
$S_1^1, S_4^0$	$S_2^0, S_5^0$	$S_3^1, S_6^0$	no
$S_1^1, S_6^0$	$S_2^0, S_3^1$	$S_4^0, S_5^1$	no
$S_1^1, S_6^0$	$S_2^0, S_5^0$	$S_3^1, S_4^0$	yes

The atomic polar decomposition narrows the search space from 15 possible axiom bindings to 6. But there is still a gap between 6 candidate axiom bindings and the two among them associated with proofs.

To analyze the situation more closely, we must consider the information provided by the body of a proof between the endsequent and the axiom leaves. These inference steps involve three kinds of information:

**type structure:** the antecedent- and succedent-rules for the operators  $/$  and  $\backslash$  are associated with different ordering conditions;

**premises:** the logical rules are either 1-premise rules or 2-premise rules; the 2-premise rules partition the subformulas of the conclusion into two premise sequents, with the result that the atomic subformulas of one of the premise sequents cannot be paired in an axiom leaf with any of the atomic subformulas of the other. (It is for this reason that there are no proofs of  $S_1^1/S_2^0, S_3^1, S_4^0 \backslash S_5^1 \vdash S_6^0$  which pair  $S_1$  and  $S_2$  or pair  $S_4$  and  $S_5$ , even though the elements of these pairs are of opposite polarity; and this restriction determines the set of possible axiom-leaf pairings.) Thus, application of the 2-premise rules backwards from the endsequent is the engine of decomposition leading from complex sequents to axiom leaves whose antecedent and succedent consist of single atomic types.

**structural rules:** structural rules determine the resources made available by sequents themselves; following Dunn [10] and Belnap [1], we regard structural rules as one component of the structural properties of sequents, the other being the mode of formula composition embodied in Gentzen's comma.

In the sections to follow, we discuss a system of term labelling which expresses all these properties—a system introduced in [29]. We view this work as an extension of research by Roorda [30] and Moortgat [24, 25]. As in Moortgat's work, the term labels may be interpreted as terms in the linguistic dimension corresponding to phonological (here: orthographical) structure. Accordingly, we call such terms  $\varphi$ -terms. But while Moortgat's  $\varphi$ -terms are elements of an algebra  $\langle S, + \rangle$ , the  $\varphi$ -terms introduced here are higher-order terms. This has a number of consequences: it allows a simplification of the type system, since directionality need no longer be coded in the type system; it allows comparison with the  $\lambda$ -term models for **LP** and **L** of van Benthem [2] and Buszkowski [7]; finally, it yields new forms of controlled deduction of linguistic interest.

### 3 Schematic labelled deduction

A number of different kinds of models have been proposed for substructural logics like **L** and **LP**, a diversity which no doubt depends on the many kinds of structures which depend on a product with residuations. Prominent among these are *algebraic models* [9, 6, 3] and  *$\lambda$ -term models* [2, 7, 3]. In the standard case, one begins with a type language and a consequence relation, interprets the types and the relation in a model structure, and investigates such questions as soundness and completeness. Our interest here is in using model-theoretic information to label types, in line with the slogan ‘bringing semantics into syntax’. The model-theoretic information involved in the systems of labelled deduction described below involves a combination of algebraic models and  $\lambda$ -term models: that is, we label types with a pair of terms, one term representing the form of linguistic expressions, the other representing aspects of linguistic interpretation. Term structure is tied to type-deduction in a straightforward way. And thus relative to lexical assumptions, a consequence relation implicitly defines a relation between linguistic form and linguistic interpretation. Our focus here is on systems of type deduction with a commutative, associative product  $\otimes$  and its residual  $\overset{\otimes}{\rightarrow}$ , and the variety of consequence relations that arise by labelling this simple type system with different kinds of terms.

In what follows, our aim is a definition of *labelled type* and a concomitant notion of *labelled deduction*. We begin with types, and then introduce a notion of *typed  $\varphi$ -terms*. A labelled type will be a pairing of a type and an appropriate term.

*Types.* Given a non-empty set  $At$  of atomic types, the set  $Typ$  is the smallest set containing  $At$  and closed under the binary operations  $\overset{\otimes}{\rightarrow}$  and  $\otimes$ .

*$\varphi$ -types.* There is a single atomic  $\varphi$ -type: the type  $s$  (for ‘string’); if  $A$  and  $B$  are  $\varphi$ -types, so are  $A \rightarrow B$  and  $\langle A, B \rangle$ ; that’s all.

*$\varphi$ -terms.* The  $\varphi$ -terms arise by constructing a suitable collection of *higher order terms* over an algebraic structure. A convenient structure for our purposes is a freely-generated monoid  $\langle \mathcal{S}^*, \cdot, 1 \rangle$ , with  $\mathcal{S}^*$  the closure of a nonempty set  $\mathcal{S}$  under an associate binary operation  $\cdot$  which has a two-sided unit 1. All of the elements of  $\mathcal{S}^*$  will be regarded as terms of type  $s$  and the operation  $\cdot$  will be of type  $\langle s, s \rangle \rightarrow s$ . We adjoin, for each type  $\alpha$ , denumerably many variables  $x_0^\alpha, x_1^\alpha, \dots, x_n^\alpha, \dots$ . In order for the terms defined to be useful for the control of type inference, they will be *linear*: no variable will occur more than once in any term (and indeed, in any sequent). To keep track of the resources represented in each term, we define the terms as follows, simultaneously defining for each term  $t$  the set  $FV(t)$  of *free variables* of  $t$  and the multiset  $FA(t)$  of free atoms of  $t$ .

1. each element  $t \in \mathcal{S}$  is a  $\varphi$ -term of type  $s$ ;  $FA(t) = t$ ;  $FV(t) = \emptyset$ ;
2. each variable  $x^\alpha$  is a  $\varphi$ -term of type  $\alpha$ ;  $FA(x) = x = FV(x)$ ;
3. if  $A$  and  $B$  are  $\varphi$ -terms of type  $s$  with  $FV(A) \cap FV(B) = \emptyset$ , then  $A \cdot B$  is a  $\varphi$ -term of type  $s$ ;  $FA(A \cdot B) = FA(A) \sqcup FA(B)$ ,  $FV(A \cdot B) = FV(A) \cup FV(B)$ ;
4. for all  $\varphi$ -terms  $A, B, C$  of type  $s$ ,  $(A \cdot B) \cdot C = A \cdot (B \cdot C)$  (if defined);
5. there is a distinguished  $\varphi$ -term constant 1 of type  $s$  with the property that for all  $\varphi$ -terms  $A$  of type  $s$ ,  $1 \cdot A = A = A \cdot 1$ ;  $FA(1) = \emptyset = FV(1)$ ;

6. if  $A$  is a  $\varphi$ -term of type  $\alpha \rightarrow \beta$  and  $B$  is a  $\varphi$ -term of type  $\beta$ , with  $FV(A) \cap FV(B) = \emptyset$ , then  $(AB)$  is a  $\varphi$ -term of type  $\beta$ ;  $FA((AB)) = FA(A) \sqcup FA(B)$ ,  $FV((AB)) = FV(A) \cup FV(B)$ ;
7. if  $x$  is a variable  $\varphi$ -term of type  $\alpha$  and  $A$  is a  $\varphi$ -term of type  $\beta$  with  $x \in FV(A)$ , then  $(\lambda x.A)$  is a  $\varphi$ -term of type  $\alpha \rightarrow \beta$ ;  $FA((\lambda x.A)) = FA(A) \setminus \{x\}$ ;  $FV((\lambda x.A)) = FV(A) \setminus \{x\}$ ;
8. if  $A$  and  $B$  are  $\varphi$ -terms of types  $\alpha$  and  $\beta$ , respectively, with  $FV(A) \cap FV(B) = \emptyset$ , then  $\langle A, B \rangle$  is a  $\varphi$ -term of type  $\langle \alpha, \beta \rangle$ ;  $FA(\langle A, B \rangle) = FA(A) \sqcup FA(B)$ ,  $FV(\langle A, B \rangle) = FV(A) \cup FV(B)$ ;
9. if  $A = \langle B, C \rangle$  is a  $\varphi$ -term of type  $\langle \alpha, \beta \rangle$ , then  $\pi_1 A = B$  and  $\pi_2 A = C$  are  $\varphi$ -terms of types  $\alpha$  and  $\beta$ , respectively;  $FA(\pi_1 A) = FA(B)$ ,  $FA(\pi_2 A) = FA(C)$ ,  $FV(\pi_1 A) = FV(B)$ ,  $FV(\pi_2 A) = FV(C)$ .

Note that for this set of terms, if a term of the form  $\lambda x.\alpha$  is well-formed, then  $\alpha$  contains exactly 1 occurrence of the variable  $x$ . We call this property *linearity*. Substitution and  $\beta$ -conversion are defined as usual [15]: we write  $\alpha \triangleright \alpha'$  when  $\alpha'$  is the result of removing one or more  $\beta$ -redexes from  $\alpha$  by  $\beta$ -reduction, or replacing one or more subterms of the form  $\pi_1 \langle \gamma, \delta \rangle$  or of the form  $\pi_2 \langle \gamma, \delta \rangle$  by  $\gamma$  or  $\delta$  (respectively), or both. If  $\alpha \triangleright \alpha'$  and there is no  $\alpha''$  for which  $\alpha' \triangleright \alpha''$ , we say that  $\alpha'$  is a term in normal form and write  $\alpha \triangleright^* \alpha'$ .

*Labelled types.* A labelled type is an element of  $Typ$  associated with a  $\varphi$ -term and a  $\lambda$ -term. This association is regulated by type-compatibility functions  $\tau_\varphi : Typ \rightarrow \varphi$ -types and  $\tau_\lambda : Typ \rightarrow \lambda$ -types. For each atom  $A \in Typ$ , we have  $\tau_\varphi(A) = s$ ; and we assume that  $\tau_\lambda(A)$  is defined. Inductively, we have:

$$\begin{aligned} \tau_\varphi(A \rightarrow B) &= \tau_\varphi(A) \rightarrow \tau_\varphi(B) \\ \tau_\lambda(A \rightarrow B) &= \tau_\lambda(A) \rightarrow \tau_\lambda(B) \\ \tau_\varphi(A \otimes B) &= \langle \tau_\varphi(A), \tau_\varphi(B) \rangle \\ \tau_\lambda(A \otimes B) &= \langle \tau_\lambda(A), \tau_\lambda(B) \rangle \end{aligned}$$

We display labelled types in the form  $\varphi : T : \lambda$ , with  $T \in Typ$ ,  $\varphi$  a  $\varphi$ -term of type  $\tau_\varphi(T)$  and  $\lambda$  a  $\lambda$ -term of type  $\tau_\lambda(T)$ .

### 3.1 *Sequent calculus*

A sequent presentation may be given as follows. In view of the role of term labels, we take a sequent to be a triple  $\langle \Gamma, A, X \rangle$ , written  $\Gamma \vdash_X A$ , with  $\Gamma$  a non-empty multiset of labelled types,  $A$  a single labelled type, and  $X$  the set of free variables in the terms of  $\Gamma$  and  $A$ . For simplicity, we represent only the  $\varphi$ -term labels, since the dynamics of the labels system are the same in both labelling systems.  $FV(t)$  denotes the free variables in the term  $t$ . We use  $X \sqcup Y$  to represent the union of disjoint sets  $X$  and  $Y$ , and  $\Gamma, \Delta$  denotes the multiset union of the multisets  $\Gamma$  and  $\Delta$ . By abuse of notation, we allow a single variable or a single type to also denote the singleton set containing just the variable or the singleton multiset containing just the type.

#### Axiom

$$\frac{A \in \mathcal{A}, u \triangleright^* u'}{u : A \vdash_{FV(u')} u' : A}$$

**Logical Rules**

$$\begin{array}{c}
 \frac{\Delta \vdash_X u : A \quad \Gamma, t(u) : B \vdash_{Y \sqcup FV(u)} v : C}{\Gamma, \Delta, t : A \xrightarrow{\otimes} B \vdash_{X \sqcup Y} v : C} \quad L^{\otimes} \qquad \frac{\Gamma, u : A \vdash_{X \sqcup u} t : B}{\Gamma \vdash_X \lambda u. t : A \xrightarrow{\otimes} B : \lambda u' : t'} \quad R^{\otimes} \\
 \\
 \frac{\Gamma, \pi_1 u : A, \pi_2 u : B \vdash_X v : C}{\Gamma, u : A \otimes B \vdash_X v : C} \quad L^{\otimes} \qquad \frac{\Gamma \vdash_X u : A \quad \Delta \vdash_Y v : B}{\Gamma, \Delta \vdash_{X \sqcup Y} \langle u, v \rangle : A \otimes B} \quad R^{\otimes}
 \end{array}$$

**Structural Rules**

$$\text{Cut} \frac{\Gamma \vdash_X u : A \quad u : A, \Delta \vdash_{Y \sqcup FV(u)} v : B}{\Gamma, \Delta \vdash_{X \sqcup Y} v : B}$$

REMARK 3.1

The following two sequents are instances of axiom-leaves, with the second illustrating the role of normalization in the identity axiom:

$$\begin{array}{l}
 \text{jones} : np : j \vdash \text{jones} : np : j \\
 ((\lambda u. u \text{ walks})j) : s : ((\lambda u'. \text{walk}'(u'))j) \vdash \text{jones walks} : s : \text{walk}'(j)
 \end{array}$$

If we regard these two axioms as the left and right premisses of an instance of the inference figure  $L \xrightarrow{\otimes}$ , the conclusion takes the form:

$$\text{jones} : np : j, (\lambda u. u \text{ walks}) : np \xrightarrow{\otimes} s : (\lambda u'. \text{walk}'(u')) \vdash \text{jones walks} : s : \text{walk}'(j)$$

This illustrates a simple proof.

In the absence of the term normalization enforced here in axiom instances (or alternatively, in the right-hand premise of the rule  $L \xrightarrow{\otimes}$ ), the terms of the succedent type of the endsequent could not be shown to be normal.

Note also that in the premise of the rule  $R \xrightarrow{\otimes}$ , the term variable  $u$  is required to be fresh: in effect, we think of  $u$  as a variable in the conclusion of this rule and as a parameter in the premise, where it behaves like a uniquely-occurring constant.

Because the  $\varphi$ -terms of this system are based on a monoid, and the type system is based on **LP**, we call this system **mon:LP**.

Following Lambek [18], Gentzen's Cut Elimination Theorem can be adapted to show that any sequent provable **mon:LP** can be proved without the use of Cut.

A second useful insight into the properties of this system comes from adapting an observation of van Benthem [2]. Call a sequent  $\Gamma \vdash t : A : t$  *term-linear* if  $FA(\Gamma) = FA(A)$ . Then we have the theorem, straightforwardly proved by induction:

THEOREM 3.2

Any **mon:LP**-valid sequent is term-linear.

### 3.2 Decomposition proofs

We now consider another presentation, which is the application here of the atomic polar decomposition introduced by Roorda [30] and Moortgat [24]. Our discussion draws heavily on the fuller exposition in [29].

As before, we begin by assigning polarities to sequent types, polarity 1 to antecedent types, polarity 0 to succedent types. We then proceed as follows, where the table below indicates how terms are to be treated—in particular, when fresh term variables are to be introduced:

complex polar type	decomposed polar form
$\text{Term} : A \otimes B^0 : \text{Term}$	$[\text{Term} \mapsto \langle \pi_1 \text{Term}, \pi_2 \text{Term}, \rangle$ $\text{Term} \mapsto \langle \pi_1 \text{Term}, \pi_2 \text{Term}, \rangle,$ $\pi_1 \text{Term} : A^0 : \pi_1 \text{Term}, \pi_2 \text{Term} : B^0 : \pi_2 \text{Term}]$
$\text{Term} : A \otimes B^1 : \text{Term}$	$[\text{Term} \mapsto \langle \pi_1 \text{Term}, \pi_2 \text{Term}, \rangle$ $\text{Term} \mapsto \langle \pi_1 \text{Term}, \pi_2 \text{Term}, \rangle,$ $\pi_1 \text{Term} : A^1 : \pi_1 \text{Term}, \pi_2 \text{Term} : B^1 : \pi_2 \text{Term}]$
$\text{Term} : A \overset{\otimes}{\rightarrow} B^0 : \text{Term}$	$[x, x \text{ fresh}, \text{Term} = \lambda x. \text{Term}', x @ \text{Term}',$ $\text{Term} = \lambda x. \text{Term}', x @ \text{Term}',$ $x : A^1 : x, \text{Term}' : B^0 : \text{Term}']$
$\text{Term} : A \rightarrow B^1 : \text{Term}$	$[x, x \text{ fresh}, \text{Term} = \lambda x. \text{Term}', x @ \text{Term}',$ $\text{Term} = \lambda x. \text{Term}', x @ \text{Term}',$ $x : A^1 : x, \text{Term}' : B^0 : \text{Term}']$

Iteration of these steps leads to a multiset of atomic polar types and an associated set of inclusions relating terms. Such a set is said to be *paired and anchored* if there is a matching substitution  $\theta$  of term variables associated with types of polarity 0 with ground terms or parametric variables which is consistent with term inclusion relations and  $\beta$ -conversion and which extends to a map from terms to terms in the standard way, together with a bijection  $\mu$  from labelled types of polarity 1 to labelled types of polarity 0 which, when  $\mu(t : A^1 : t) = u : B^0 : u$ , satisfies the conditions:

$$A = B, \quad \theta(t) = \theta(u), \quad \theta(t) = \theta(u)$$

The following two theorems relate these notions to sequent proofs:

**THEOREM 3.3**

If a sequent  $\Sigma$  is provable in **mon:LP**, then the atomic polar decomposition of  $\Sigma$  is paired and anchored.

**THEOREM 3.4**

If the atomic polar decomposition of a term-linear sequent  $\Sigma$  is paired and anchored, then  $\theta(\Sigma)$  is provable in **mon:LP**.

These assertions are proved in [29]. For a penetrating discussion of the issues involving matching vs. unification of term labels, see Morrill's paper [27].

We noted earlier a correspondence between axiom-leaf pairings and distinct Curry–Howard terms for sequent-style proofs. Each distinct way of pairing atomic types in axiom-leaves corresponds to a different paired anchoring of the atomic polar decomposition. Thus, to find every reading of a provable sequent in **mon:LP**, we need only determine every such paired anchoring. We illustrate this property below.

## 4 Grammatical applications

Complex expressions arise as succedent formulas of valid proofs from lexical assumptions—assumptions of the sort exemplified below:

$$\begin{aligned} \text{smith} &: np : s \\ \text{jones} &: np : j \\ \lambda x \lambda y . y \text{ questioned } x &: np \otimes (np \otimes s) : \lambda x \lambda y . \text{question}'(x)(y) \\ \lambda z \lambda P . (P \text{ (every } \cdot z)) &: n \otimes ((np \otimes s) \otimes s) : \lambda Q \lambda P . \forall (Q)(P) \\ \lambda z \lambda P . (P \text{ (a } \cdot z)) &: n \otimes ((np \otimes s) \otimes s) : \lambda Q \lambda P . \exists (Q)(P) \\ \text{dean} &: n : \text{dean}' \\ \text{student} &: n : \text{student}' \end{aligned}$$

Consider now the sequent  $\langle \Gamma, A, \emptyset \rangle$  with  $\Gamma$  and  $A$  defined as follows:

$$\begin{aligned} \Gamma &= [ \text{smith} : np : s, \\ &\quad \text{jones} : np : j, \\ &\quad \lambda x \lambda y . y \text{ questioned } x : np \otimes (np \otimes s) : \lambda x \lambda y . \text{question}'(x)(y) \\ &] \\ A &= \text{smith questioned jones} : s : \text{question}'(j)(s) \end{aligned}$$

To show that this sequent is valid by decomposition, we assign polarity 1 to the elements of  $\Gamma$  and polarity 0 to  $A$ , yielding the following multiset of polar labelled types:

$$\begin{aligned} [ \text{smith} &: np^1 : s, \\ \text{jones} &: np^1 : j, \\ \lambda x \lambda y . y \text{ questioned } x &: np \otimes (np \otimes s)^1 : \lambda x \lambda y . \text{question}'(x)(y), \\ \text{smith questioned jones} &: s^0 : \text{question}'(j)(s) \\ ] \end{aligned}$$

The only complex polar labelled type in this multiset is the type corresponding to the transitive verb. Its atomic polar decomposition is displayed below:

$$\begin{aligned} [ x &: np^0 : x, \\ y &: np^0 : y, \\ y \text{ questioned } x &: s^1 : \text{question}'(x)(y) \\ ] \end{aligned}$$

Replacing the lexical assumption corresponding to the transitive verb with its atomic polar decomposition yields the atomic polar decomposition of the sequent  $\langle \Gamma, A, \emptyset \rangle$ :

$$\begin{aligned} [ \text{smith} &: np^1 : s, \\ \text{jones} &: np^1 : j, \\ x &: np^0 : x, \\ y &: np^0 : y, \\ y \text{ questioned } x &: s^1 : \text{question}'(x)(y) \\ \text{smith questioned jones} &: s^0 : \text{question}'(j)(s) \\ ] \end{aligned}$$

We need to show that this multiset of labelled signed types is paired and anchored. This can be done in precisely one way, with the following matching substitution  $\theta$ :

$$\begin{aligned} \theta : \quad & x \mapsto \text{jones}, \\ & x \mapsto j, \\ & y \mapsto \text{smith}, \\ & y \mapsto s \end{aligned}$$

#### 4.1 *A quantificational example*

Quantificational examples such as a dean questioned every student provide a more interesting challenge. The lexical assumptions corresponding to the determiners every and a can be decomposed as follows (choosing fresh variables where required and writing  $t[u]$  to indicate that  $t$  contains a free occurrence of  $u$ ):

$$\begin{aligned} & \lambda z \lambda P. (P (\text{every} \cdot z)) : n \overset{\otimes}{\rightarrow} ((np \overset{\otimes}{\rightarrow} s) \overset{\otimes}{\rightarrow} s) : \lambda Z \lambda P. \forall (Z)(P) \\ \rightsquigarrow [ & \lambda z \lambda P. (P (\text{every} \cdot z)) : n \overset{\otimes}{\rightarrow} ((np \overset{\otimes}{\rightarrow} s) \overset{\otimes}{\rightarrow} s)^1 : \lambda Z \lambda P. \forall (Z)(P)] \\ \rightsquigarrow [ & z : n^0 : Z, \\ & \lambda P. (P (\text{every} \cdot z)) : (np \overset{\otimes}{\rightarrow} s) \overset{\otimes}{\rightarrow} s^1 : \lambda P. \forall (Z)(P)] \\ \rightsquigarrow [ & z : n^0 : Z, \\ & P : np \overset{\otimes}{\rightarrow} s^0 : P, \\ & (P (\text{every} \cdot z)) : s^1 : \forall (Z)(P)] \\ \rightsquigarrow [ & z : n^0 : Z, \\ & u : np^1 : u, \\ & f[u] : s^0 : \varphi[u], \\ & (\lambda u. (f[u])(\text{every} \cdot z)) : s^1 : \forall (Z)(\lambda u. (\varphi[u]))] \\ & \lambda r \lambda S. (S (a \cdot r)) : n \overset{\otimes}{\rightarrow} ((np \overset{\otimes}{\rightarrow} s) \overset{\otimes}{\rightarrow} s) : \lambda R \lambda S. \exists (R)(S) \\ \rightsquigarrow [ & \lambda r \lambda S. (S (a \cdot r)) : n \overset{\otimes}{\rightarrow} ((np \overset{\otimes}{\rightarrow} s) \overset{\otimes}{\rightarrow} s)^1 : \lambda R \lambda S. \exists (R)(S)] \\ \rightsquigarrow [ & r : n^0 : R, \\ & \lambda S. (S (a \cdot r)) : (np \overset{\otimes}{\rightarrow} s) \overset{\otimes}{\rightarrow} s^1 : \lambda S. \exists (R)(S)] \\ \rightsquigarrow [ & r : n^0 : R, \\ & S : np \overset{\otimes}{\rightarrow} s^0 : S, \\ & (S (a \cdot r)) : s^1 : \exists (R)(S)] \\ \rightsquigarrow [ & r : n^0 : R, \\ & v : np^1 : v, \\ & g[v] : s^0 : \psi[v], \\ & (\lambda v. (g[v])(a \cdot r)) : s^1 : \exists (R)(\lambda v. \psi[v])] \end{aligned}$$

Consider now the sequent  $\langle \Delta, B, \emptyset \rangle$ , with  $\Delta$  and  $B$  as follows:

$$\begin{aligned} \Delta = [ & \lambda r \lambda S. (S(a \cdot r)) : n \overset{\otimes}{\rightarrow} ((np \overset{\otimes}{\rightarrow} s) \overset{\otimes}{\rightarrow} s) : \lambda R \lambda S. \exists (R)(S), \\ & \lambda x \lambda y. y \text{ questioned } x : np \overset{\otimes}{\rightarrow} (np \overset{\otimes}{\rightarrow} s)^1 : \lambda x \lambda y. \text{question}'(x)(y), \\ & \lambda z \lambda P. (P(\text{every} \cdot z)) : n \overset{\otimes}{\rightarrow} ((np \overset{\otimes}{\rightarrow} s) \overset{\otimes}{\rightarrow} s) : \lambda Z \lambda P. \forall (Z)(P), \\ & \text{dean} : n : \text{dean}', \\ & \text{student} : n : \text{student}' \\ & ] \\ B = & \text{ a dean questioned every student} : s : \text{Term} \end{aligned}$$

The interpretation of the succedent  $B$  is represented by the variable  $Term$ . The different anchors of this variable correspond to the range of interpretations assigned

by the Curry–Howard principles to the sequent proof.

To determine the values of *Term*, we first replace the sequent  $\langle \Delta, B, \emptyset \rangle$  by its atomic polar decomposition:

$$\begin{array}{l}
 [ \quad r : n^0 : R, \\
 \quad v : np^1 : v, \\
 \quad g[v] : s^0 : \psi[v], \\
 \quad (\lambda v. (g[v])(a \cdot r)) : s^1 : \exists (R)(\lambda v. \psi[v]), \\
 \quad x : np^0 : x, \\
 \quad y : np^0 : y, \\
 \quad y \text{ questioned } x : s^1 : \textit{question}'(x)(y), \\
 \quad z : n^0 : Z, \\
 \quad u : np^1 : u, \\
 \quad f[u] : s^0 : \varphi[u], \\
 \quad (\lambda u. (f[u])(\textit{every} \cdot z)) : s^1 : \forall (Z)(\lambda u. (\varphi[u])), \\
 \quad \textit{dean} : n^1 : \textit{dean}', \\
 \quad \textit{student} : n^1 : \textit{student}', \\
 \quad \textit{a dean questioned every student} : s^0 : \textit{Term} \\
 ]
 \end{array}$$

Certain bindings are forced by the properties of the string-term in the last line, which represents the goal of the goal-directed proof search. Thus, since we have a *dean* . . . , then the determiner *a* must combine with the noun *dean*, requiring  $r \mapsto \textit{dean}$  and  $R \mapsto \textit{dean}'$ . Similarly,  $z \mapsto \textit{student}$  and  $Z \mapsto \textit{student}'$  are forced. Moreover, since we have a *dean questioned* . . . and . . . *questioned every student*, we must pair  $y : np^0 : y$  with  $v : np^1 : v$ , on the one hand, and  $x : np^0 : x$  with  $u : np^1 : u$ , on the other. This leaves open only the bindings of the sentential types, displayed below, in a way reflecting these forced bindings:

$$\begin{array}{l}
 (\lambda v. (g[v])(a \cdot r)) : s^1 : \exists (R)(\lambda v. \psi[v]), \\
 g[v] : s^0 : \psi[v], \\
 v \text{ questioned } u : s^1 : \textit{question}'(u)(v), \\
 f[u] : s^0 : \varphi[u], \\
 (\lambda u. (f[u])(\textit{every} \cdot z)) : s^1 : \forall (Z)(\lambda u. (\varphi[u])), \\
 \textit{a dean questioned every student} : s^0 : \textit{Term}
 \end{array}$$

We cannot identify the first of these with  $g[v] : s^0 : \psi[v]$ , because even though they involve the same atomic type (*s*) of opposite polarity, the interpretive term of the first (namely,  $\exists (R)(\lambda v. \psi[v])$ ) properly contains the interpretive term of the second. For the same reason, we cannot identify the labelled type containing *every* with  $f[u] : s^0 : \varphi[u]$ . We can, however, identify  $(\lambda v. (g[v])(a \cdot r)) : s^1 : \exists (R)(\lambda v. \psi[v])$  and  $f[u] : s^0 : \varphi[u]$ . On this assumption, we must have the following pairings:

$$\begin{array}{l}
 (\lambda v. (g[v])(a \cdot r)) : s^1 : \exists (R)(\lambda v. \psi[v]), \\
 \simeq f[u] : s^0 : \varphi[u], \\
 \\
 g[v] : s^0 : \psi[v], \\
 \simeq v \text{ questioned } u : s^1 : \textit{question}'(u)(v), \\
 \\
 (\lambda u. (f[u])(\textit{every} \cdot z)) : s^1 : \forall (Z)(\lambda u. (\varphi[u])), \\
 \simeq \textit{a dean questioned every student} : s^0 : \textit{Term}
 \end{array}$$

Thus, we have:

$$\begin{array}{lcl}
\mathbf{g}[v] & \mapsto & v \text{ questioned } u \\
\psi[v] & \mapsto & \mathit{question}'(u)(v) \\
\mathbf{f}[u] & \mapsto & a \text{ dean questioned } u \\
\varphi[u] & \mapsto & \exists (\mathit{dean}')(\lambda v . \mathit{question}'(u)(v)) \\
\mathit{Term} & \mapsto & \forall (\mathit{student}')(\lambda u . \exists (\mathit{dean}')(\lambda v . \mathit{question}'(u)(v)))
\end{array}$$

On this solution, the universal quantifier has wide scope.

Alternatively, we can identify  $(\lambda v . (\mathbf{g}[v])(a \cdot r) : s^1 : \exists (R)(\lambda v . \psi[v]))$  with the succedent type a dean questioned every student :  $s^0 : \mathit{Term}$ . This forces:

$$\begin{array}{l}
(\lambda v . (\mathbf{g}[v])(a \cdot r) : s^1 : \exists (R)(\lambda v . \psi[v])), \\
\simeq \text{ a dean questioned every student : } s^0 : \mathit{Term}
\end{array}$$

$$\begin{array}{l}
\mathbf{f}[u] : s^0 : \varphi[u], \\
\simeq v \text{ questioned } u : s^1 : \mathit{question}'(u)(v),
\end{array}$$

$$\begin{array}{l}
(\lambda u . (\mathbf{f}[u])(\text{every} \cdot z) : s^1 : \forall (Z)(\lambda u . (\varphi[u])), \\
\simeq \mathbf{g}[v] : s^0 : \psi[v]
\end{array}$$

On this solution, we have:

$$\mathit{Term} \mapsto \exists (\mathit{dean}')(\lambda v . \forall (\mathit{student}')(\lambda u . \mathit{question}'(u)(v)))$$

No other solutions are possible.

An empirically interesting aspect of this system is the fact that quantifier scope ambiguities arise intrinsically as a form of proof indeterminacy. There is no special scoping rule, such as Montague's rule of Quantifying In, or the rule of Quantifier Raising found in the GB literature. Moreover, there is no need in this system to associate proper names with lifted types, to assign multiple types to quantifiers, or to assign higher-order types to simple transitive verb (together with meaning postulates). Finally, within this framework, extraction and infixation operations are assimilated to the properties of residuals, so the type constructors proposed by Moortgat [22, 23] are represented here without extra machinery. For further discussion, see [29].

## 5 Parametric variation

An important thread in recent work on the Curry–Howard morphism is the role of variation in the properties of term-labels [2, 7, 12]. What are the parameters of variation in the present case?

One obvious parameter involves the choice of the algebraic basis of the  $\varphi$ -terms. In **mon:LP**, this basis consists of a freely-generated monoid. Other choices are possible, however.

### 5.1 LP

If we choose a commutative, non-idempotent monoid  $M$ , then our terms are higher-order terms constructed over multisets of string atoms. Call this system **com-mon:LP**. This system has obvious affinities with **LP**, since the  $\varphi$ -terms impose no constraints on type deduction.

5.2 *L*

Call the atomic constants of a **mon:LP**  $\varphi$ -term its *string support*. We can restrict the terms to just those whose string support is a connected sub-term. This allows such first-order terms as  $\lambda x. x \text{ walks}$ ,  $\lambda x. \lambda y. \lambda z. x \text{ gave } y z$  (where  $x, y, z$  are variables of type  $s$ ), but not  $\lambda x. \lambda z. \lambda y. x \text{ gave } y z$ , since application of this term to two ground terms  $a, b$  yields  $\lambda y. a \text{ gave } y b$ , whose string support  $a \text{ gave } b$  is not a connected sub-term. In these terms, we may discern a reflex of the directional distinction in **L** between types of the form  $s/np$  and  $np\s$ . But in higher-order types, directional distinctions become indiscernible. Let  $P$  be a  $\varphi$ -term variable of type  $s \rightarrow s$ : a term of the form  $\lambda P. P$  (every student) combines equally with left-looking functors like  $\lambda x. x \text{ walks}$  and right-looking functors like  $\lambda x. \text{ smith contacted } x$ . Thus, in order to embed a system like **L** into a set of labelled terms, either the types must preserve the directional information carried by  $/$  and  $\backslash$  or the terms themselves must carry comparable directional information.

There is some linguistic interest to these observations. From the point of view of 3-dimensional labelled deduction (or just 2-dimensional, if we consider only the labelling of types with  $\varphi$ -terms), reconstructing directionally-sensitive systems like **L** requires the introduction of new type information. In this sense, **mon:LP** is a simpler system within this space of possibilities. (The importance of the space of evaluation here is analogous to whether one regards phonological segments as atomic units (like alphabetic letters) or as bundles of distinctive features: the latter possibility brings with it an intrinsic notion of natural phonological class, whereas the former does not. We may apply this analogy in the present setting by examining properties of term-labels and the way they are assigned to types.)

5.3 *NL*

Suppose we replace the monoid  $S^*$  with a freely-generated non-associative, non-commutative, non-idempotent groupoid. By itself, this change does not basically change the system of labelled deduction: terms simply carry along a fixed binary-branching structure. What is needed is to restrict abstraction in the definition of terms in an appropriate way. We must allow such terms as  $\lambda x. (x \cdot t)$  (corresponding to  $B\backslash A$ ),  $\lambda x. (t \cdot x)$  (corresponding to  $A/B$ ), and  $\lambda x. \lambda y. (y \cdot (x \cdot t))$  (corresponding to  $C\backslash(B\backslash A)$ ). The latter type suggests the restriction: the number of string brackets between an occurrence of an abstraction operator and the variable it binds minus the number of intervening abstraction operators equals 1. This restriction suffices to block the derivability of composition, since in the sequent displayed below, the succedent term does not conform to it:

$$\lambda x. (fx) : A \overset{\otimes}{\rightarrow} B, \lambda y. (gy) : B \overset{\otimes}{\rightarrow} C \vdash \lambda z. (g(fz)) : A \overset{\otimes}{\rightarrow} C$$

Call this system **grp:LP**, since it is based on a non-associative groupoid. This system also represents a more sophisticated system than **mon:LP**, since the terms associated with the conclusion of deductive steps need to be checked to ascertain whether or not they conform to the depth requirements. This contrasts with the situation in **mon:LP**, where every term derived is well-formed. (Issues of this kind have been insightfully analyzed by Blackburn & de Rijke in [4].)

#### 5.4 Grammatical data structures as terms

Another source of possible term-structures can be found in the various data structures of contemporary grammatical practice. We touch on this possibility only briefly.

##### 5.4.1 TAG operations

If we choose a non-commutative, non-idempotent, non-associative groupoid  $G$ , then the underlying algebra of atomic  $\varphi$ -terms is an algebra of trees; application of a first-order  $\varphi$ -term to an atomic  $\varphi$ -term is substitution of the argument for the variable bound by the  $\lambda$ -operator in the first-order term. Moreover, it is possible to define the tree-adjunction operation of the TAG formalism using second-order terms of the form:

$$\lambda P.\lambda x.(P(Q(x)))$$

where  $P$  represents the ‘auxiliary tree’ (in the terminology of [17]).

##### 5.4.2 LFG structures

In **LFG** [5], there are links between a phrase-structure representation of an expression (its *c-structure*) and the representation of the grammatical functions of its various components (its *f-structure*). We can think of these structures as forming a double system of term labels. This would allow us to connect work in **LFG** with research in the categorial tradition.

## 6 Conclusion

This paper illustrates some of the possibilities inherent in systems of labelled deduction with higher-order terms. I have tried to emphasize the insights that arise when we examine type inference in a setting where types are labelled by two sets of terms—one representing aspects of linguistic form; the other, aspects of linguistic interpretation. This leads to new possibilities for the division of linguistic labor from the general point of categorial grammar: in particular, information concerning the order of expressions can be removed from the type system and relegated to the system of  $\varphi$ -terms. This simplification opens up the way for an account of quantifier-scope ambiguities as a form of proof indeterminacy and for a preliminary inspection of a new family of labelled deductive systems.

Future directions include a more intensive study of the connection between the class of systems studied here and the algebraic model theory of substructural logics and further exploration of multimodal systems in which the atomic types are labelled with terms from an algebra with a family of operations, of the kind explored in Moortgat’s paper in this volume.

## Acknowledgments

This paper has benefited from comments from Mark Hepple and discussion with Glyn Morrill. I owe a special debt to Michael Moortgat for conversations on all the topics touched on in this paper, from which I have learned a great deal. Errors of mine are

not to be attributed to them, of course. I also would like to thank Ruth Kempson for organizing the London workshop on proof theory and natural language in March 1994 and to Merrill Garrett of the University of Arizona Cognitive Science Program for supporting my attendance at it.

## References

- [1] N. Belnap. 1982. Display logic. *J. Phil. Logic* **11**, 375-417.
- [2] J. van Benthem. 1988. The semantics of variety in categorial grammar. In W. Buszkowski, et al., eds. *Categorial Grammar*. 37-55. Amsterdam/Philadelphia, John Benjamins.
- [3] J. van Benthem. 1991. *Language in action*. North-Holland, Amsterdam.
- [4] P. Blackburn & M. de Rijke. to appear. Zooming in, zooming out. In J. Seligman & D. Westerstahl, eds., *Language, Logic and Computation: The Moraga Proceedings*. CSLI, Stanford.
- [5] J. W. Bresnan, ed. 1982. *The mental representation of grammatical relations*. Cambridge, Mass., MIT Press.
- [6] W. Buszkowski. 1986. Completeness results for Lambek syntactic calculus. *Zeitschr. f. math. Logik und Grundlagen d. Math.* **32** 13-28.
- [7] W. Buszkowski. 1987. The logic of types. In J. Szrednicki, ed. *Initiatives in Logic*. Dordrecht, M. Nijhoff.
- [8] M. Calcagno. 1994. A sign-based extension to the Lambek Calculus for discontinuous constituency. Manuscript, Dept. of Linguistics, Ohio State University.
- [9] K. Došen. 1992. A brief survey of frames for the Lambek calculus. *Zeitschr. f. math. Logik und Grundlagen d. Math.* **38** 179-187.
- [10] J. M. Dunn. 1973. A 'Gentzen system' for positive relevant implication. *JSL* **38**, 356-357.
- [11] D. Gabbay. 1991. *Labelled Deductive Systems*. ms., Imperial College, London.
- [12] D. Gabbay & R. de Queiroz. 1992. Extending the Curry-Howard interpretation to linear, relevant and other resource logics. *JSL*. **57.4**.1319-1365.
- [13] G. Gentzen. 1934-5. Untersuchungen über das logische Schliessen, *Mathematische Zeitschrift* **39**, pp. 176-210, 405-431. (English translation in M. E. Szabo, ed., *The Collected Papers of Gerhard Gentzen*, North-Holland, Amsterdam.)
- [14] J.-Y. Girard, Y. Lafont, and P. Taylor. 1989. *Proofs and Types*. Cambridge Tracts in Theoretical Computer Science 7. Cambridge University Press, Cambridge.
- [15] J. R. Hindley & J. P. Seldin. *Introduction to Combinators and  $\lambda$ -Calculus*. London Mathematical Society Student Texts 1. Cambridge University Press, Cambridge.
- [16] W. A. Howard. 1980. The formulae-as-types notion of construction. In J.R. Hindley and J.P. Seldin, eds., *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism*, Academic Press.
- [17] A. Joshi. 1987. An introduction to Tree Adjoining Grammars. In A. Manaster-Ramer, ed., *Mathematics of Language*, Amsterdam/Philadelphia, John Benjamins.
- [18] J. Lambek. 1958. The mathematics of sentence structure. *American Mathematical Monthly* **65**, 154-170.
- [19] J. Lambek. 1961. On the calculus of syntactic types. In R. Jakobson, ed., *Structure of language and its mathematical aspects*, American Mathematical Society, Providence, 166-178.
- [20] J. Lambek. 1994. From categorial grammar to bilinear logic. K. Došen & P. Schroeder-Heister, eds. *Substructural Logic*. pp. 207-237. Oxford University Press, Oxford.
- [21] R. Montague. The proper treatment of quantification in English. In [31]. pp. 247-270.
- [22] M. Moortgat. 1988. *Categorial Investigations: Logical and Linguistic Aspects of the Lambek Calculus*. Foris, Dordrecht.
- [23] M. Moortgat. 1992. The logic of discontinuous type constructors. In W. Sijsma & A. van Horck, eds., *Discontinuous Constituency*. Mouton de Gruyter, Berlin.
- [24] M. Moortgat. 1990. Unambiguous proof representations for the Lambek calculus. In *Proceedings of the 7th Amsterdam Colloquium*, ITLI, University of Amsterdam.
- [25] M. Moortgat. 1991. Labelled deductive systems for categorial theorem proving. *Proceedings of the 8th Amsterdam Colloquium*. Universiteit van Amsterdam.

- [26] M. Moortgat & R. T. Oehrle. 1993. *Lecture Notes on Categorical Grammar*. European Summer School in Logic, Language, and Information. Faculdade de Letras, Universidade de Lisboa, Portugal.
- [27] G. Morrill. 1994. Clausal proof nets and discontinuity. Paper presented at the London Workshop on Proof Theory and Linguistic Analysis.
- [28] R. T. Oehrle. 1988. Multi-dimensional compositional functions as a basis for grammatical analysis. In R. T. Oehrle, E. Bach, and D. Wheeler, eds., *Categorical Grammars and Natural Language Structures*, D. Reidel, Dordrecht, pp. 349-389.
- [29] R. T. Oehrle. in press. Term-labelled categorial type systems. *Linguistics & Philosophy*.
- [30] D. Roorda. 1991. *Resource Logics: Proof-theoretical Investigations*. Ph.D. thesis, Faculteit van Wiskunde en Informatica, Universiteit van Amsterdam.
- [31] R. Thomason. 1974. *Formal Philosophy: Selected Papers of Richard Montague*, Yale University Press, New Haven.
- [32] A. S. Troelstra. 1992. *Lectures on linear logic*. CSLI Lecture Notes No. 29. Stanford, CSLI.
- [33] L. A. Wallen. *Automated proof search in non-classical logics: efficient matrix methods for modal and intuitionistic logics*. MIT Press, Cambridge, Mass.

Received 4 January 1995

—