Categorial Deductions and Structural Operations

Glyn Morrill, Neil Leslie, Mark Hepple and Guy Barry

Categorial grammar is an approach to language description in which expressions are classified according to recursively defined categories. The combination of expressions is governed not by specific linguistic rules but by general logical inference mechanisms, so that all the information about the combining potential of a word resides in its lexical categorization. Various systems of inference for categorial grammar have been proposed, in particular the 'syntactic calculus' of Lambek (1958). However, use of the Lambek calculus for linguistic work has generally been rather limited. There appear to be two main reasons for this: the notations most commonly used can sometimes obscure the structure of proofs and fail to clearly convey linguistic structure, and the calculus as it stands is apparently not powerful enough to describe many phenomena encountered in natural language.

In this paper we suggest ways of dealing with both these deficiencies. Firstly, we reformulate the Lambek calculus using proof figures based on the 'natural deduction' notation commonly used for derivations in logic. Natural deduction is generally regarded as the most economical and comprehensible system for working on proofs by hand, and we suggest that the same advantages hold for a similar presentation of categorial derivations. Secondly, we introduce devices for the characterization of word order variation, iteration and optionality, features of natural language which the Lambek calculus does not capture with the desired sensitivity and generality. This is achieved by means of operators called *structural modalities*, derived from the usual *structural rules* found in logic.

1 Fregean Analysis

The point of departure for categorial grammar can be seen as Frege's position that there are certain 'complete expressions' which are the primary bearers of meaning, and that the meanings of 'incomplete expressions' (including words) are derivative, being their contribution to the meanings of the expressions in which they occur. Frege took proper names and sentences to be the complete expressions. Consider a language containing some proper names {"John", "Mary", ...} and some sentences {"John walks", "Mary walks", ..., "John likes John", ...}; such sets of linguistic objects will be called *categories*, and will be indexed by *types*, in this case NP and S respectively.[1] Now, the significance of "walks" is that it occurs with an NP to form an S, and it is attributed with a meaning which is a function from NP meanings into the meanings of the S's it forms with those NP's; the category of such objects can be indexed

We would like to thank Martin Pickering and Pete Whitelock for comments and discussion relating to this work. The authors were respectively supported by SERC Postdoctoral Fellowship B/ITF/206; ESPRIT Project 393 and Cognitive Science/HCI Research Initiative 89/CS01 and 89/CS25; ESRC Research Studentship C00428722003; SERC Research Studentship 88306971.

[1] Thus a type is a symbol denoting a category. There are alternative terminologies, e.g. use of *category* to mean category symbol, or *type* to mean a domain, rather than a symbol indexing a domain.

by the type NP → S, and we say that (following the Fregean procedure) type NP → S is *assigned* to "walks". Similarly, "likes John", "likes Mary", ... are assigned type NP → S, and consequently "likes" is assigned type NP → (NP → S), etc.

Note that on this view compositionality is methodological rather than empirical: the meanings of parts are *defined* in terms of the meanings of wholes. Note also that the semantic functions we build up in this way fall within a simple type hierarchy; Russell's theory of types, prompted by the paradoxes, was a major influence on Leśniewski's 'semantic categories', the immediate precursor of Ajdukiewicz and categorial grammar.

The above system for semantic type-assignment can be extended to one which is simultaneously a syntactic type-assignment, i.e. one in which the types by which incomplete expressions are classified encode the manner in which they constitute subparts of other expressions. This can be achieved by the use of a *directional* system, in which the undirected implication $Y \rightarrow X$ is replaced by two directed implications X/Y ('X over Y') and $Y\backslash X$ ('Y under X'), representing respectively incomplete expressions that combine with a following or preceding expression of type Y to form an expression of type X.[2] Thus for example "walks" forms an S when preceded by an NP, so is assigned type NP\S, and "likes" forms an NP\S when followed by an NP, so is assigned type (NP\S)/NP.

Linguistic categories are not limited to finite inhabitation, e.g. in general languages contain an infinite number of sentences. However, the set of words from which the objects in those categories are constructed is usually presumed to be a finite set, known as the *vocabulary*. The task of a grammar is to finitely describe the membership of (possibly) infinite categories, and the above type-theoretic perspective suggests a procedure for doing this. By establishing an initial type assignment relation between the vocabulary and types (a *lexicon*), a full type assignment between expressions and types may be determined, according to a general system of type inference. In the next section we shall present such a system of inference for the directed types described above, the calculus **L** of Lambek (1958), and discuss properties and linguistic applications of that system.

2 The Lambek Calculus L

2.1 Preliminaries

Given a set of primitive types, the set of *bidirectional types* is defined as shown in (1).

(1) a. If X is a primitive type
then X is a type.

b. If X and Y are types
then X/Y and $Y\backslash X$ are types.

Let us assume as basic categories prepositional phrases, noun phrases, sentences, and common nouns, indexed by primitive types PP, NP, S, and N respectively. By the principles outlined above, we may assign types to words as follows:

[2] In the alternative notation used by, for example, Steedman, these types would be written X/Y, $X\backslash Y$ respectively.

(2)

for	:= PP/NP
John, Mary	:= NP
likes	:= (NP\S)/NP
man	:= N
the	:= NP/N
thinks	:= (NP\S)/S
votes	:= (NP\S)/PP
who	:= (N\N)/(S/NP)

We assume that words have (at least) two components, form (represented by italics) and meaning (represented by boldface). For example, the word "for" will be taken to have form denoted by *for* and meaning denoted by **for**.

2.2 Proof Figures

We shall present the rules of **L** by means of *proof figures*, based on Prawitz' (1965) systems of 'natural deduction'. Natural deduction was developed by Gentzen (1936) to reflect the natural process of mathematical reasoning in which one uses a number of *inference rules* to justify a single *conclusion* on the basis of a number of propositions, called *assumptions*. During a proof one may *temporarily* make a new assumption if one of the rules licenses the subsequent withdrawal of this assumption. The rule is said to *discharge* the assumption. The conclusion is said to *depend* on the undischarged assumptions, which are called the *hypotheses* of the proof.

A proof is usually represented as a tree with the assumptions as leaves and the conclusion at the root. Finding a proof is then seen as the task of filling this tree in, and the inference rules as operations on the partially completed tree. One can write the inference rules out as such operations, but as these are rather unwieldy it is more usual to present the rules in a more compact form as operations from a set of subproofs (the *premises*) to a conclusion, as follows:

(3)

$$\frac{\begin{array}{ccccccc} \vdots & & \vdots & [Y_1]^i & & [Y_n]^i & \\ X_1 & \cdots & X_m & X_{m+1} & \cdots & X_{m+n} & \end{array}}{Z}\,\text{Rule}^i$$

This states that a proof of Z can be obtained from proofs of $X_1, \ldots, X_{m+n}$ by discharging appropriate occurrences of assumptions $Y_1, \ldots, Y_n$. The use of square brackets around an assumption indicates its discharge. The index i is used to disambiguate proofs, where there may be an uncertainty as to which rule has discharged which assumption.

As propositions are represented by formulas in logic, so linguistic categories are represented by type formulas in the Lambek calculus. The left-to-right order of types indicates the order in which the forms of subexpressions are to be concatenated to give a composite expression derived by the proof. Thus we must take note of the order and place of occurrence of the premises of the rules in the proof figures for **L**. There is also a problem with the presentation of the rules in the compact notation as some of the rules will be written *as if* they had a number of conclusions, as follows:

(4) $$\dfrac{\begin{array}{ccc}\vdots & & \vdots \\ X_1 & \cdots & X_m\end{array}}{\begin{array}{ccc}Y_1 & \cdots & Y_n\end{array}}\text{Rule}$$

This rule should be seen as a shorthand for:

(5) $$\dfrac{\begin{array}{ccc}\vdots & & \vdots \\ \dfrac{\begin{array}{ccc}X_1 & \cdots & X_m\end{array}}{\begin{array}{ccc}Y_1 & \cdots & Y_n\end{array}}\text{Rule} & & \\ \vdots \quad \vdots & & \vdots \quad \vdots\end{array}}{Z}$$

If the rules are viewed in this way it will be seen that they do not violate the single conclusion nature of the figures.[3]

As with standard natural deduction, for each connective there is an *introduction* rule which states how a type containing that connective may be derived, and an *elimination* rule which states how a type containing that connective may be consumed. We shall follow the historical development of the calculus by giving the elimination rules before the introduction rules.

The elimination rule for / states that a proof of type X/Y followed by a proof of type Y yields a proof of type X. Similarly the elimination rule for \ states that a proof of type $Y\backslash X$ preceded by a proof of type Y yields a proof of type X. Using the notation above, we may write these rules as follows:

(6) a. $$\dfrac{\begin{array}{cc}\vdots & \vdots \\ X/Y & Y\end{array}}{X}/\text{E}$$ b. $$\dfrac{\begin{array}{cc}\vdots & \vdots \\ Y & Y\backslash X\end{array}}{X}\backslash\text{E}$$

The meaning of the composite expression is given by the functional application of the meaning of the *functor* expression (i.e. the one of type X/Y or $Y\backslash X$) to the meaning of the *argument* expression (i.e. the one of type Y). We represent function application by juxtaposition, so that **likes John** means **likes** applied to **John**.

The subsystem of **L** using only the rules /E and \E corresponds to the categorial calculus of Ajdukiewicz (1935) and Bar-Hillel (1953), and is sometimes known as the **AB** calculus. Within this calculus, we may derive "Mary likes John" as a sentence as follows:

(7) $$\dfrac{\dfrac{\text{Mary}}{\text{NP}} \quad \dfrac{\dfrac{\text{likes}}{\text{(NP\S)/NP}} \quad \dfrac{\text{John}}{\text{NP}}}{\text{NP\S}}/\text{E}}{\text{S}}\backslash\text{E}$$

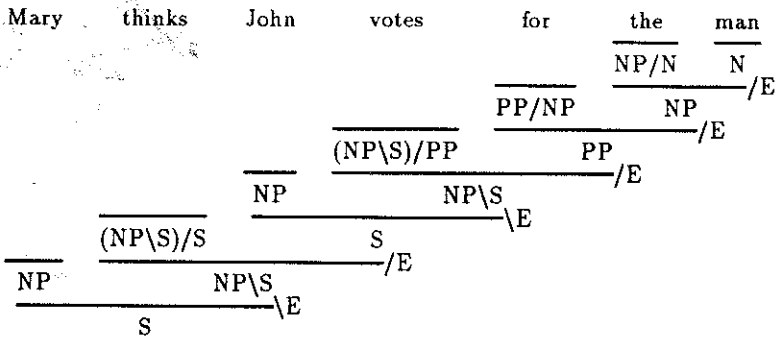
The meaning of the sentence is read off the proof by interpreting the /E and \E inferences as function application, giving the following:

(8) **(likes John) Mary**

Similarly, the sentence "Mary thinks John votes for the man" may be derived thus:

[3]Notice that these rules mean that the proof figures are not trees, but directed acyclic graphs.

(9)



From this we can read off the meaning representation:

(10) **(thinks ((votes (for (the man))) John)) Mary**

The introduction rule for / states that where the rightmost assumption in a proof of the type X is of type Y, that assumption may be discharged to give a proof of the type X/Y. Similarly, the introduction rule for \ states that where the leftmost assumption in a proof of the type X is of type Y, that assumption may be discharged to give a proof of the type $Y\backslash X$. Using the notation above, we may write these rules as follows:

(11) a. $\dfrac{\begin{array}{c}[Y]^n \\ \vdots \\ X\end{array}}{X/Y}/\mathrm{I}^n$ b. $\dfrac{\begin{array}{c}[Y]^n \\ \vdots \\ X\end{array}}{Y\backslash X}\backslash\mathrm{I}^n$

Note however that this notation does not embody the conditions that have been stated, namely that in /I Y is the rightmost undischarged assumption in the proof of X, and in \I Y is the leftmost undischarged assumption in the proof of X. In addition, the Lambek calculus carries the condition that in both /I and \I the sole assumption in a proof cannot be withdrawn, so that no types are assigned to the empty string.[4]

In the introduction rules, the meaning of the result is given by *functional abstraction* over the meaning of the discharged assumption, which can be represented by a variable of the appropriate type. For example, the functional abstraction of [(**likes** x) **Mary**] over the variable x, written λx[(**likes** x) **Mary**], is the function that gives (**likes John**) **Mary** when applied to **John**, (**likes** (**the man**)) **Mary** when applied to **the man**, and so on. This general relation between abstraction, application and substitution is expressed by the law of *β-equality*:

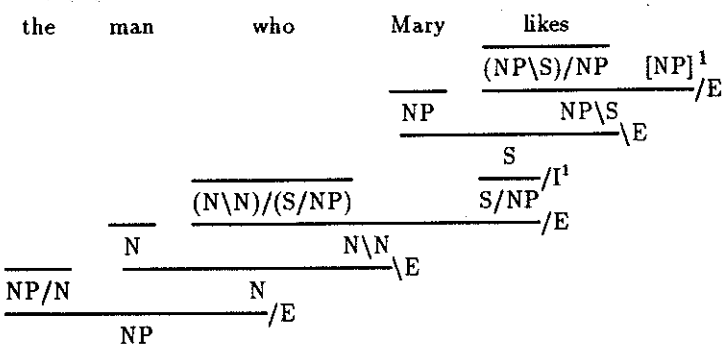
(12) $(\lambda y[\alpha])\beta = \alpha[\beta/y]$

This is read: '$(\lambda y[\alpha])\beta$ is equal to α with β substituted for y'. The rules in (11) are analogous to the usual natural deduction rule of *conditionalization*, except that the latter allows withdrawal of any number of assumptions, in any position. The semantic algebra defined by the above set of inference rules corresponds to the subset of the typed lambda-calculus in which each binder binds exactly one variable (van Benthem 1983). We shall refer to this as the 'single-bind' lambda-calculus. This correspondence between proofs in **L** and meaning representations in the single-bind lambda-calculus can be seen as a version of the Curry-Howard cor-

[4] However, this condition is not an essential feature of the logic (cf. subsection 3.1).

respondence between intuitionistic implicational deduction and full lambda-calculus (Curry and Feys 1958; Howard 1969).

The /I and \I rules are commonly used in constructions that are standardly assumed to involve 'empty categories'. For example, the non-canonical constituent "Mary likes" can be derived as of type S/NP, and hence assignment of type (N\N)/(S/NP) to an object relative pronoun allows analysis of relativization as follows (cf. Ades and Steedman 1982):

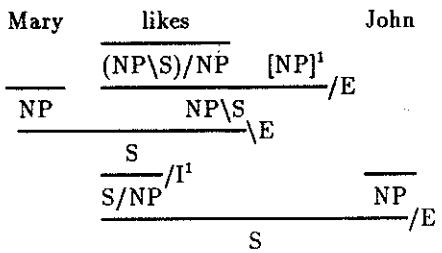
(13)


Again, the meaning of the string can be read off the above proof by interpreting /I and \I as functional abstraction, giving the following:

(14) **the (who** (λx[**(likes** x) **Mary**]) **man)**

Note that this mechanism is only powerful enough to allow constructions where the extraction site is clause-peripheral; for non-peripheral extraction, and multiple extraction such as with parasitic gaps, we appear to need an extended logic. Meeting such requirements is part of the concern of section 4 below.

The use of the /I and \I rules in conjunction with the /E and \E rules means that it is possible to give more than one proof for a single reading of a string. For example, compare the derivation of "Mary likes John" in (7), and the corresponding lambda-term in (8), with the derivation in (15) and the lambda-term in (16):

(15)


(16) (λx[**(likes** x) **Mary**]) **John**

By the definition in (12), the terms in (8) and (16) are β-equal, and thus have the same meaning; the proofs in (7) and (15) are thus equivalent. This property of *derivational equivalence* (also termed 'spurious ambiguity') is common to many versions of categorial grammar, and can cause problems for the design of efficient parsing algorithms, where the usual aim is to find each reading of a string once and only once. Such problems can however be overcome by defining a notion of normal form for proofs (and their corresponding terms) in such a way

that each equivalence class of proofs contains a unique normal form (e.g. Hepple and Morrill 1989). One advantage of the present formulation is that it offers a particularly simple way of doing this.

2.3 Normal Forms

In order to define *normal forms* for both **L** and the single-bind lambda-calculus, we will first define the notions of *contraction* and *reduction*. A contraction schema $R \triangleright C$ consists of a particular pattern R within proofs or terms (the *redex*) and an equal and simpler pattern C (the *contractum*). A reduction consists of a series of contractions, each replacing an occurrence of a redex by its contractum. A normal form is a proof or term on which no contractions are possible.

We define the following contraction schemas: *weak contraction* for proofs, and *β-contraction* for the corresponding lambda-terms.

(17) a.
$$\dfrac{\dfrac{\overset{[Y]^n}{\vdots}\ X}{X/Y}/\mathrm{I}^n \quad Y}{X}/\mathrm{E} \quad \triangleright \quad \overset{Y}{\vdots}\ X$$

b.
$$\dfrac{Y \quad \dfrac{\overset{[Y]^n}{\vdots}\ X}{Y\backslash X}\backslash \mathrm{I}^n}{X}\backslash \mathrm{E} \quad \triangleright \quad \overset{Y}{\vdots}\ X$$

(18) $(\lambda y[\alpha])\beta \triangleright \alpha[\beta/y]$

From (12) we see that β-reduction preserves meaning according to the standard functional interpretation of typed lambda-calculus. Therefore the corresponding weak reduction preserves the semantic functional interpretation of the proof; in addition it preserves the syntactic string interpretation since the redex and contractum contain the same leaves in the same order.

For example, the proof in (15) weakly contracts to the proof in (7), and correspondingly the term in (16) β-contracts to the term in (8). The results of these contractions cannot be further contracted and so are the respective results of reduction to *weak normal form* and *β-normal form.*

Weak contraction and β-contraction strictly decrease the sizes of proofs and terms, e.g. the number of symbols in a contractum is always less than that in a redex.[5] Thus there is *strong normalization* with respect to these reductions: every proof (term) reduces to a weak normal form (β-normal form) in a finite number of steps. This has as a corollary (*normalization*) that every proof (term) has a normal form, so that normal forms are *fully* representative: every proof (term) is equal to one in normal form. Since reductions preserve interpretations, an interpretation of a normal form will always be the same as that of the original proof (term). Thus restricting the search to just such proofs addresses the problem of derivational equivalence, while preserving generality in that all interpretations are found.

The single-bind proofs (lambda-terms) appear to straightforwardly inherit the property of intuitionistic proofs (full lambda-terms) that if a proof (term) M reduces to two proofs (terms) N_1, N_2, then there is a proof (term) N_3 to which both N_1 and N_2 reduce. It follows

[5] This follows immediately from the single-bind character of the languages; if there were multiple binding, contraction could involve duplication of subparts substituted and so could increase size.

that a proof (term) cannot reduce to more than one normal form, and so every proof (term) has a *unique* normal form. This is known as the *Church-Rosser* property.

If all normal forms were non-equal, restriction of search to normal forms would be optimal in that each equivalence class has a single representative within the narrowed search space. However, the standard extensional functional interpretation of typed lambda-calculus also satisfies the following law of *η-equality*:

(19) $\lambda y[\alpha(y)] = \alpha$
(y not free in α)

Accordingly there is a second form of contraction for each of **L** and the single-bind lambda-calculus, which we shall term *strong contraction* and *η-contraction* respectively. These are as follows:

(20) a.
$$\dfrac{\dfrac{\vdots}{X/Y} \quad \dfrac{[Y]^n}{\vdots}/\mathrm{E}}{\dfrac{X}{X/Y}/\mathrm{I}^n} \quad \triangleright \quad \dfrac{\vdots}{X/Y}$$

b.
$$\dfrac{\dfrac{\vdots}{[Y]^n} \quad \dfrac{Y\backslash X}{\vdots}\backslash\mathrm{E}}{\dfrac{X}{Y\backslash X}\backslash\mathrm{I}^n} \quad \triangleright \quad \dfrac{\vdots}{Y\backslash X}$$

(21) $\lambda y[\alpha(y)] \triangleright \alpha$

A proof (term) is said to be in *strong normal form* (*βη-normal form*) if there are no weak or strong contractions (*β-* or *η-* contractions) possible on it. Since strong reductions (*βη-*reductions) also preserve interpretations, it is sufficient to find all and only the strong normal proofs for a given string and a given result type in order to find all possible readings in that type.

A notion of contraction creates a partitioning into normal and non-normal proofs. For the purposes of proof search a structural characterization of just the normal proofs can be exploited (see e.g. König 1989; Hepple 1990). Weak normal proofs can be specified as follows. Suppose we define a *branch* of a proof in **L** as starting with any hypothesis and tracing down until it hits either the conclusion type (a main branch) or the argument type of an elimination inference (a side branch). Then in a weak normal proof no branch can contain an introduction inference followed by an elimination inference. It follows that each branch must consist of a sequence of (zero or more) eliminations followed by a sequence of (zero or more) introductions. However, a similar structural characterization of strong normal proofs is harder to formulate.

2.4 Sequent Calculus

The above notation for proofs is convenient for their presentation on the page, but for the development of proof-search algorithms it is more convenient to use Gentzen's (1936) *sequent calculus* notation, where the objects reasoned about are *sequents* corresponding to entire natural deduction proofs. A sequent calculus consists of axioms and inference rules; axioms state that certain primitive sequents are valid, and inference rules allow new valid sequents to be derived from old ones. A sequent calculus proof may thus be seen as the instructions for the construction of a natural deduction proof. Lambek (1958), in his original presentation of the calculus, used a sequent calculus notation in order to establish its decidability.

We shall use $\Gamma, \Delta, \Phi \ldots$ to stand for sequences of types, and $X, Y, Z \ldots$ to stand for single occurrences of a type. A (single-conclusioned) sequent is a statement $\Gamma \Rightarrow X$; it asserts that there is a deduction of X depending on the hypotheses Γ. In Lambek calculus both the order of the types in Γ and the number of occurrences of each type are relevant, and there is a restriction that Γ must be non-empty.

We shall present the sequent rules in *refinement* notation to emphasize their application to proof search. A rule such as (22) states that Z follows from Γ if X follows from Δ and Y follows from Φ. It does not matter which order we solve the sub-problems in, so long as we solve them all.

(22) $\Gamma \Rightarrow Z$ [Rule]
$\Delta \Rightarrow X$
$\Phi \Rightarrow Y$

In standard sequent systems there is a single schematic axiom saying that every type yields itself, and for each connective there are two inference rules; a *left rule*, which allows us to derive a sequent where the connective newly appears on the left-hand side of the arrow, and a *right rule*, which allows us to derive a sequent where the connective newly appears on the right-hand side of the arrow. The Gentzen-style formulation of **L** follows this pattern:

(23) $X \Rightarrow X$ [Ax]

$$\frac{\Gamma\ X/Y\ \Delta\ \Phi \Rightarrow Z \quad [/\mathrm{L}]}{\Delta \Rightarrow Y,\ \Gamma\ X\ \Phi \Rightarrow Z} \qquad \frac{\Gamma\ \Delta\ Y\backslash X\ \Phi \Rightarrow Z \quad [\backslash\mathrm{L}]}{\Delta \Rightarrow Y,\ \Gamma\ X\ \Phi \Rightarrow Z}$$

$$\frac{\Gamma \Rightarrow X/Y \quad [/\mathrm{R}]}{\Gamma\ Y \Rightarrow X} \qquad \frac{\Gamma \Rightarrow Y\backslash X \quad [\backslash\mathrm{R}]}{Y\ \Gamma \Rightarrow X}$$

There is a close analogy between the use of left rules in sequent calculus and elimination rules in natural deduction, and similarly between right rules in sequent calculus and introduction rules in natural deduction. For example, compare the following sequent proof with the proof in (13):

(24)
NP/N N (N\N)/(S/NP) NP (NP\S)/NP ⇒ NP [/L]
N (N\N)/(S/NP) NP (NP\S)/NP ⇒ N [/L]
NP (NP\S)/NP ⇒ S/NP [/R]
NP (NP\S)/NP NP ⇒ S [/L]
NP ⇒ NP [Ax]
NP NP\S ⇒ S [\L]
NP ⇒ NP [Ax]
S⇒S [Ax]
N N\N ⇒ N [\L]
N ⇒ N [Ax]
N ⇒ N [Ax]
NP ⇒ NP [Ax]

L may be formulated with one additional rule, the *Cut rule*, which combines two proofs by identifying the conclusion of one with one of the premises of the other:

$$(25)\quad \frac{\Delta\ \Gamma\ \Phi \Rightarrow Y \quad [\text{Cut}]}{\begin{array}{l}\Gamma \Rightarrow X \\ \Delta\ X\ \Phi \Rightarrow Y\end{array}}$$

Lambek shows that the sequent formulation of **L** has the property of *Cut-elimination*, i.e. every **L** proof can be transformed into an **L** proof with the same end-sequent, in which no Cuts occur. Cut-elimination in sequent calculus corresponds (approximately) to normalization in natural deduction (cf. the corresponding result for intuitionistic logic; see also Moortgat 1990a). The Cut-free formulation provides a decision procedure. A step in the search for a proof of a sequent can only be an axiom matching, or one of a finite number of possible rule applications, these creating strictly simpler subgoals since each rule application removes a connective.

Moortgat (1988) shows how the types in sequents in **L** may be annotated with lambda-terms to indicate their semantic content; the resulting terms can be shown to be the same as those obtained in the natural deduction style formulation.

3 A Structural Hierarchy of Logics

The rest of this paper will be concerned with the development of mechanisms extending **L** to a calculus with greater linguistic applicability, in particular addressing the problems of word order variation, iteration and optionality. In this section we describe how **L** can be extended to yield a structural hierarchy of logics, and in the next section we refine these extensions in a manner suited to linguistic description.

From a logical perspective, **L** can be thought of as the weakest member of a hierarchy of implicational type-inference logics, differing in the amount of freedom allowed in the use of assumptions; **L** takes into account the order of assumptions and the number of occurrences of each, but we may choose to ignore these factors to a greater or lesser extent. This 'categorial hierarchy' is discussed in e.g. van Benthem (1987), Moortgat (1988), Ono (1988) and Wansing (1989). It consists of **L** followed by what are essentially the implicational fragments of linear, relevance and intuitionistic logic.

Sequent calculus differentiates between levels in the hierarchy by means of *structural rules* explicitly manipulating lists of assumptions. In natural deduction style formulations, on the other hand, structural operations are implicit in the treatment of occurrences of assumptions and the conditions on their discharge. In the presentations below we shall break with common practice by including explicit structural rules in proof figures; this will enable a closer comparison with the structural operators to be discussed in the next section.

3.1 L and Sequential Logic

L as it stands forms a 'sequential logic', which is a subset of *non-commutative linear logic* (see Girard 1989). The single-conclusioned, implicational fragment of non-commutative linear logic corresponds exactly to **L**, except that proofs from no premises are permissible in the former.

3.2 LP and Linear Logic

The 'Lambek-van Benthem' calculus **LP** (van Benthem 1983, 1986) is like **L** except that the order of premises is irrelevant. We may implement this by adding a structural rule of *permutation* to the natural deduction proof figures:

(26) $$\dfrac{\begin{array}{cc}\vdots & \vdots \\ X & Y\end{array}}{\begin{array}{cc}Y & X\end{array}}\,\mathrm{P}$$

This is the first appearance of a rule that is not single-conclusioned. As noted earlier, such a rule may not form the last inference in a proof, since complete proofs must have a single conclusion type.

Addition of the permutation rule means that the directional implicational types X/Y, $Y\backslash X$ are equivalent, since they are mutually derivable:

(27) $$\dfrac{\dfrac{\dfrac{[Y]^1 \quad X/Y}{X/Y \quad Y}\,\mathrm{P}}{X}\,/\mathrm{E}}{Y\backslash X}\,\backslash\mathrm{I}^1 \qquad \dfrac{\dfrac{\dfrac{Y\backslash X \quad [Y]^1}{Y \quad Y\backslash X}\,\mathrm{P}}{X}\,\backslash\mathrm{E}}{X/Y}\,/\mathrm{I}^1$$

For the rest of this section, therefore, we need use only a single implicational type; we choose $Y\backslash X$, and notate it in the more conventional fashion as $Y\rightarrow X$. We repeat the elimination and introduction rules here:

(28) $$\dfrac{\begin{array}{cc}\vdots & \vdots \\ Y & Y\rightarrow X\end{array}}{X}\,\rightarrow\mathrm{E} \qquad \dfrac{\begin{array}{c}[Y]^n \\ \vdots \\ X\end{array}}{Y\rightarrow X}\,\rightarrow\mathrm{I}^n$$

Without loss of generality, rule applications can be kept subject to the previous ordering conditions for \. For example, we may derive the transition $X\rightarrow(Y\rightarrow Z) \Rightarrow Y\rightarrow(X\rightarrow Z)$ in **LP** as follows:

(29) $$\dfrac{\dfrac{\dfrac{\dfrac{[X]^1 \quad [Y]^2}{Y \quad X}\,\mathrm{P} \quad X\rightarrow(Y\rightarrow Z)}{Y\rightarrow Z}\,\rightarrow\mathrm{E}}{Z}\,\rightarrow\mathrm{E}}{X\rightarrow Z}\,\rightarrow\mathrm{I}^1}{Y\rightarrow(X\rightarrow Z)}\,\rightarrow\mathrm{I}^2$$

However, since the order of premises is irrelevant, the logic is unaffected by allowing the argument type Y in the $\rightarrow$E rule to occur either to the left or the right of the functor type $Y\rightarrow X$, and Y to be *any* undischarged assumption above X in the $\rightarrow$I rule. This removes the need for an explicit permutation rule. In a sequent calculus presentation of **LP**, the structural rule of permutation runs as follows:

(30) $$\dfrac{\Gamma\ X\ Y\ \Delta \Rightarrow Z}{\Gamma\ Y\ X\ \Delta \Rightarrow Z}\ [\mathrm{P}]$$

LP corresponds to the single-bind lambda-calculus, and is a subsystem of Girard's (1987) *linear logic*, in which permutation of assumptions preserves derivability; the implication in **LP** is like Girard's *linear implication*.

3.3 LPC and Relevance Logic

LPC is an extension of **LP** which allows assumptions to be used more than once. This may be achieved by adding a structural rule of *contraction* to the natural deduction formulation:

(31) $$\dfrac{\begin{array}{c}\vdots\\ X\end{array}}{X\quad X}\,\mathrm{C}$$

The $\rightarrow$E and $\rightarrow$I rules are as before in this formulation. By way of example of derivation in **LPC**, the transition $X\rightarrow(X\rightarrow Y) \Rightarrow X\rightarrow Y$ is proved as follows:

(32) $$\dfrac{\dfrac{\dfrac{[X]^1}{X\quad X}\,\mathrm{C}\qquad X\rightarrow(X\rightarrow Y)}{X\rightarrow Y}\rightarrow\mathrm{E}}{Y}\rightarrow\mathrm{E}$$
$$\dfrac{Y}{X\rightarrow Y}\rightarrow\mathrm{I}^1$$

LPC could be alternatively formulated in a natural deduction style by relaxing the directionality conditions on **L** and allowing withdrawal of one or more occurrences of a conditionalized assumption. This removes the need for explicit rules of permutation and contraction. The sequent calculus formulation of the contraction rule is as follows:

(33) $$\dfrac{\Gamma\ X\ \Delta \Rightarrow Z}{\Gamma\ X\ X\ \Delta \Rightarrow Z}\ [\mathrm{C}]$$

LPC corresponds to the multiple-bind lambda-calculus without vacuous abstraction, and it shares with *relevance logic* (Anderson and Belnap 1975) the preservation of derivability under the structural operations of permutation and contraction.

3.4 LPCW and Intuitionistic Logic

Finally we may extend **LPC** to **LPCW**, where it is permissible for assumptions not to be used at all. This may be achieved by adding a structural rule of *weakening* to the natural deduction formulation:

(34) $$\dfrac{\begin{array}{c}\vdots\\ X\end{array}}{}\,\mathrm{W}$$

In fact, in order that a weakened assumption may be understood as a subpart of a proof, applications of weakening in a derivation will be represented as follows:

(35) a. $\dfrac{\begin{array}{cc}\vdots & \vdots \\ X & Y\end{array}}{Y}\,W_l$ b. $\dfrac{\begin{array}{cc}\vdots & \vdots \\ Y & X\end{array}}{Y}\,W_r$

For example, the transition $Y \Rightarrow X \rightarrow Y$ is valid in **LPCW**:

(36) $\dfrac{\dfrac{[X]^1 \quad Y}{Y}\,W_l}{X \rightarrow Y}\,I^1$

This derivation satisfies the ordering conditions on conditionalization since X is indeed the leftmost assumption dominating Y. **LPCW** could be alternatively formulated in a natural deduction style by relaxing the directionality conditions on **L** and allowing withdrawal of zero or more occurrences of a conditionalized assumption. This removes the need for explicit rules of permutation, contraction and weakening. The sequent calculus formulation of the weakening rule is as follows:

(37) $\dfrac{\Gamma\ X\ \Delta \Rightarrow Z}{\Gamma\ \Delta \Rightarrow Z}\ [W]$

LPCW corresponds to the full lambda-calculus allowing vacuous abstraction, and it shares with *intuitionistic logic* the preservation of derivability under the structural operations of permutation, contraction and weakening.

4 Structural Modalities

Freely applying structural rules are clearly not appropriate in categorial grammars for linguistic description. Permutation would imply that word order was irrelevant; contraction would imply that any word in a sentence could fulfil any number of roles; and weakening would imply that sentences could contain arbitrary words not contributing to their meaning. Nevertheless, commutable, iterable, and optional elements do arise in natural language: a relative pronoun may license a gap[6] anywhere in body of the relative clause, it may license multiple gaps (as in parasitic gap constructions), and a gap may be optionally required (perhaps as in resumptive pronoun constructions); similarly a head may allow some freedom in the location of a complement, it may allow multiple complements (as in iterated coordination), and it may optionally take a complement. This suggests that we should be able to indicate that structural operations are permissible on specific types, while still forbidding them as general inference rules.

There is a precedent for the above in (commutative) linear logic, which contains the so-called *exponentials* ! ('of course') and ? ('why not'), which bring back the structural rules of contraction and weakening in a controlled way. The behaviour of ! and ? bears a resemblance to that of the operators $\Box$ (universal modality) and $\Diamond$ (existential modality) in some modal logics. For our linguistic calculus, which has rather different applications, we shall suggest a system of operators called *structural modalities*. Corresponding to commutation, iteration and optionality, we define both universal and existential structural modalities; there are two directional versions for each of the commutation modalities.

[6] We use the term 'gap' purely descriptively.

(38)

	Universal	Existential
Commutation	$\triangleright\triangleleft$	$><$
Iteration	!	+
Optionality	‖	?

Broadly speaking the universal modalities may be said to represent 'any' (any location, any number of occurrences greater than zero, any number of occurrences less than two), and the existential modalities to represent 'some' (some location, some number of occurrences greater than zero, some number of occurrences less than two). For each modality there is an 'operational rule' which governs its behaviour; the operational rules for universal modalities are analogous to the corresponding structural rules, and those for existential modalities are analogous to inverted versions of the corresponding structural rules. We shall examine these rules in detail for each operator below.

Universal modalities have the common property that any item of a universally modal type also has the corresponding non-modal type, and the existential modalities have the property that any item of a non-modal type also has the corresponding existentially modal type. This means there are straightforward elimination rules for the universal modalities and introduction rules for the existential modalities, but the complementary rules in each case are less straightforward. Since we are not fixing here a particular semantics with respect to which we expect the logic to be complete, we shall be ignoring introduction rules for the universal modalities and elimination rules for the existential modalities.

In grammar, the universals and existentials will effect the structural operations with respect to gaps and realized elements respectively. When we look at lexical type-assignments involving these modalities, we will find that existential modalities are appropriate for describing the behaviour of lexical items that subcategorize for commutable, iterable or optional arguments; such items typically receive types of the form $\Diamond Y \rightarrow X$, where $\Diamond$ represents one of the existential modalities and $\rightarrow$ one of the implications.[7] Conversely, universal modalities are appropriate for describing the behaviour of lexical items that subcategorize for strings that are missing commutable, iterable or optional arguments; such items typically receive types of the form $(\Box Z \rightarrow Y) \rightarrow X$, where $\Box$ represents one of the universal modalities. As we shall see in the examples, it follows from these assignments that the missing introduction and elimination rules are rarely needed for linguistic purposes.

4.1 Commutation Modalities

We notate the two universal commutation modalities as $\triangleright$ (written as a prefix operator) and $\triangleleft$ (written as a postfix operator). The type $\triangleright X$ (resp. $X \triangleleft$) is assigned to an item of type X which may occur in its current position or anywhere to the right (resp. left) of its current position. This is achieved by means of the rules $\triangleright$Perm and $\triangleleft$Perm, which allow types bearing the appropriate modality to be moved to any position on the given side, and the rules $\triangleright$E and $\triangleleft$E, which allow the modality to be dropped:

(39) a. $\dfrac{\vdots \quad \vdots}{\triangleright X \quad Y} \triangleright\text{Perm}$ $\qquad$ b. $\dfrac{\vdots \quad \vdots}{Y \quad X \triangleleft} \triangleleft\text{Perm}$

$$Y \quad \triangleright X \qquad\qquad X \triangleleft \quad Y$$

[7] Here, and uniformly in what follows, unary operators bind tighter than binary ones.

(40) a. $\dfrac{\vdots\ \triangleright X}{X}\triangleright E$ b. $\dfrac{\vdots\ X\triangleleft}{X}\triangleleft E$

The above may also be expressed as sequent rules:

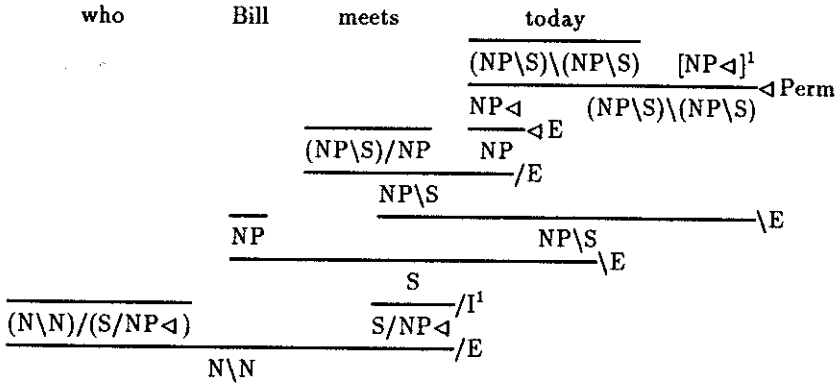
(41) $\dfrac{\Gamma\ \triangleright X\ Y\ \Delta \Rightarrow Z}{\Gamma\ Y\ \triangleright X\ \Delta \Rightarrow Z}\ [\triangleright\text{Perm}]$ $\qquad$ $\dfrac{\Gamma\ Y\ X\triangleleft\ \Delta \Rightarrow Z}{\Gamma\ X\triangleleft\ Y\ \Delta \Rightarrow Z}\ [\triangleleft\text{Perm}]$

$\dfrac{\Gamma\ \triangleright X\ \Delta \Rightarrow Y}{\Gamma\ X\ \Delta \Rightarrow Y}\ [\triangleright\text{L}]$ $\qquad$ $\dfrac{\Gamma\ X\triangleleft\ \Delta \Rightarrow Y}{\Gamma\ X\ \Delta \Rightarrow Y}\ [\triangleleft\text{L}]$

Note that the structural operation performed by the $\triangleright$Perm and $\triangleleft$Perm rules (in either formulation) is the normal structural operation of permutation.

We may use these modalities in a treatment of non-peripheral extraction, as in "the man who Bill meets today". Moortgat (1988) deals with such constructions by means of the extraction operator $\uparrow$; the type $X \uparrow Y$ is assigned to any string of type X missing a subpart of type Y somewhere within it. So assigning the type $(N\backslash N)/(S\uparrow NP)$ to object relative pronouns licenses extraction from *any* position in the body of a relative clause. We may accomplish the same in the present framework by giving relative pronouns the type $(N\backslash N)/(S/NP\triangleleft)$:

(42)



Moortgat (1990b) suggests that the type $X \uparrow Y$ may in fact be definable in terms of modalities similar to the ones above.

We notate the two existential commutability modalities as $>$ (written postfix) and $<$ (written prefix). The type $X>$ (resp. $<X$) is assigned to an item of type X either in its current position or somewhere to the right (resp. left) of its current position. This is achieved by means of the rules $>$I and $<$I, which allow the appropriate modality to be added, and the operational rules $>$Exch and $<$Exch ('exchange'), which allow types bearing the modality to be moved to any position on the given side:

(43) a. $\dfrac{\vdots\ X}{X>}>\text{I}$ b. $\dfrac{\vdots\ X}{<X}<\text{I}$

(44) a. $\dfrac{\vdots\ X> \quad \vdots\ Y}{Y \quad X>}>\text{Exch}$ b. $\dfrac{\vdots\ Y \quad \vdots\ <X}{<X \quad Y}<\text{Exch}$

The corresponding sequent rules are as follows:

(45) $\dfrac{\Gamma \Rightarrow X}{\Gamma \Rightarrow X>}\ [>\mathrm{R}]$ $\qquad$ $\dfrac{\Gamma \Rightarrow X}{\Gamma \Rightarrow <X}\ [<\mathrm{R}]$

$\dfrac{\Gamma(X\ Y) \Rightarrow Z}{\Gamma(Y\ X>) \Rightarrow Z}\ [>\mathrm{Exch}]$ $\qquad$ $\dfrac{\Gamma(Y\ X) \Rightarrow Z}{\Gamma(<X\ Y) \Rightarrow Z}\ [<\mathrm{Exch}]$

Note that the permutation operation is symmetrical and so the structural operations associated with the >Exch and <Exch rules are trivially inversions of the usual permutation operation.

By way of example a type $X/<Y$ represents elements which are functors over Ys *somewhere* to the right, but not necessarily immediately to the right. In this way we can define the 'long-distance' functors $X//Y$ and $Y\backslash\backslash X$ of Bach (1984). Typically such specifications would need to be made with reference also to the domains within which the word order variation is allowed, e.g. by using the modal treatment of boundedness introduced in Morrill (1989), but the essential idea is as illustrated by Bach's example of SOV and OSV alternation in Japanese:

(46) a. Nao-ga neko-o miru
PN-subj cat-obj sees
'Nao sees a cat'

b. neko-o Nao-ga miru
'Nao sees a cat'

(47) a.
$$\dfrac{\text{Nao-ga}\quad \dfrac{\dfrac{\text{neko-o}}{\text{NPo}}}{\text{NPo>}}>\mathrm{I}\quad \dfrac{\text{miru}}{\text{NPo>\(\backslash\)(NPs\(\backslash\)S)}}}{\dfrac{\text{NPs}\qquad\qquad \text{NPs\(\backslash\)S}}{\text{S}}\backslash\mathrm{E}}\backslash\mathrm{E}$$

b.
$$\dfrac{\dfrac{\dfrac{\dfrac{\text{neko-o}}{\text{NPo}}}{\text{NPo>}}>\mathrm{I}\quad \dfrac{\text{Nao-ga}}{\text{NPs}}}{\text{NPs}\qquad \text{NPo>}}>\mathrm{Exch}\quad \dfrac{\text{miru}}{\text{NPo>\(\backslash\)(NPs\(\backslash\)S)}}}{\text{NPs\(\backslash\)S}}\backslash\mathrm{E}}{\text{S}}\backslash\mathrm{E}$$

4.2 Iterability Modalities

The universal iterability modality is such that the type $X^{!}$ is assigned to an item from which one or more occurrences of the type X may be derived. This is achieved by means of the rule $^{!}$Con, which copies any type bearing the operator, and the rule $^{!}$E, which removes the operator.

(48) a.
$$\dfrac{\begin{array}{c}\vdots\\ X^{!}\end{array}}{X^{!}\quad X^{!}}\,{}^{!}\mathrm{Con}$$

b.
$$\dfrac{\begin{array}{c}\vdots\\ X^{!}\end{array}}{X}\,{}^{!}\mathrm{E}$$

The corresponding sequent rules are:

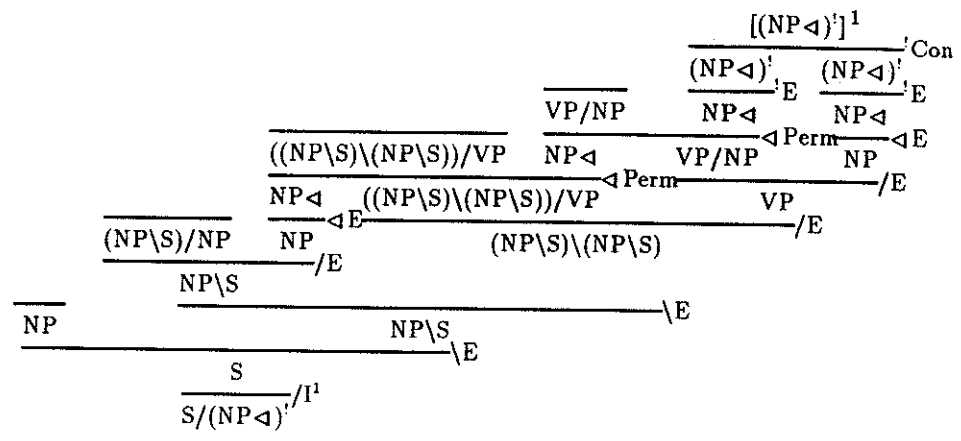
(49) $\dfrac{\Gamma\ X^{!}\ \Delta \Rightarrow Y}{\Gamma\ X^{!}\ X^{!}\ \Delta \Rightarrow Y}\ [{}^{!}\mathrm{Con}]$ $\qquad$ $\dfrac{\Gamma\ X^{!}\ \Delta \Rightarrow Y}{\Gamma\ X\ \Delta \Rightarrow Y}\ [{}^{!}\mathrm{L}]$

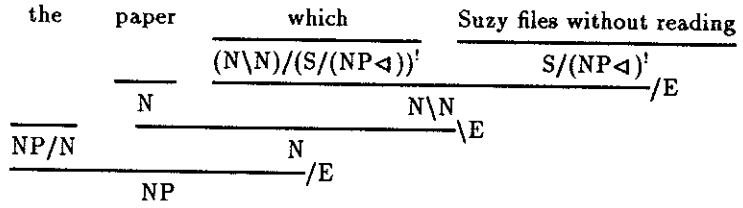
The structural operation performed by the $^{!}$Con rule is contraction. Note that the premise in the sequent formulation of this rule is more complex than the conclusion since it contains at

least an extra $^{\downarrow}$, and so the sequent formulation no longer provides a straightforward decision procedure.

We now have the apparatus for a rudimentary treatment of parasitic gaps. Assignment of the type $(N\backslash N)/(S/(NP\triangleleft)^{\downarrow})$ to a relative pronoun will allow it to fill *any number of* gaps in any positions. (This clearly overgenerates, but allows an illustration of the interaction of iteration and commutation operators.) "The paper which Suzy files without reading" can thus be derived as follows:

(50) Suzy files without reading





The existential iteration modality is such that X^{+} represents some non-zero number of items of type X. This is achieved by means of the $^{+}$I rule, which introduces the operator, and the $^{+}$Exp ('expansion') rule, which combines identical types bearing it:

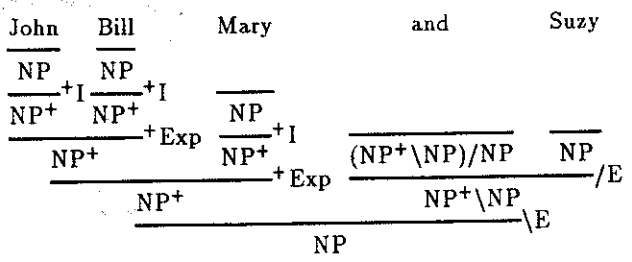
(51) a. $\dfrac{\vdots\ X}{X^{+}}\,{}^{+}\mathrm{I}$ b. $\dfrac{\vdots\ X^{+}\quad \vdots\ X^{+}}{X^{+}}\,{}^{+}\mathrm{Exp}$

Correspondingly there are the following sequent rules:

(52) $\dfrac{\Gamma \Rightarrow X^{+}}{\Gamma \Rightarrow X}\ [{}^{+}\mathrm{R}]$ $\qquad$ $\dfrac{\Gamma\ X^{+}\ X^{+}\ \Delta \Rightarrow Y}{\Gamma\ X^{+}\ \Delta \Rightarrow Y}\ [{}^{+}\mathrm{Exp}]$

The operation associated with the $^{+}$Exp rule is the inverse of the operation of contraction.

Iterated coordination may be treated by assignment of coordinators to $(X^+\backslash X)/X$:

(53) 

4.3 Optionality Modalities

The universal optionality modality is such that $X^{\|}$ is assigned to an item from which zero or one occurrences of the type X may be derived. This is achieved by means of the rules $^{\|}\text{Wkn}_l$ and $^{\|}\text{Wkn}_r$, which remove any type bearing the operator, and a standard universal modality elimination rule $^{\|}\text{E}$:

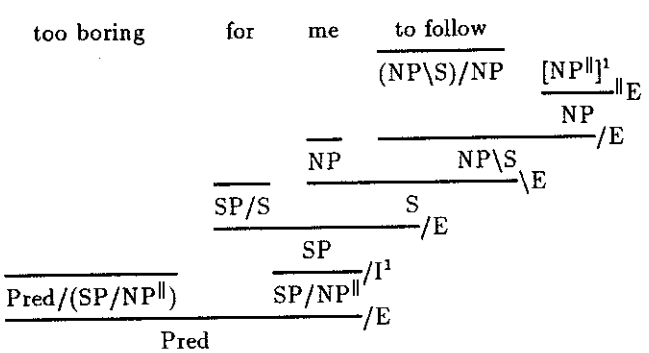
(54) a. $\dfrac{X^{\|} \quad Y}{Y}\,{}^{\|}\text{Wkn}_l$ b. $\dfrac{Y \quad X^{\|}}{Y}\,{}^{\|}\text{Wkn}_r$ c. $\dfrac{X^{\|}}{X}\,{}^{\|}\text{E}$

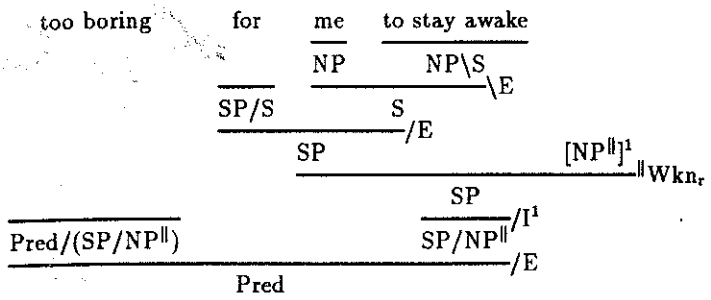
The corresponding sequent rules are:

(55) $\dfrac{\Gamma\ \Delta \Rightarrow Y}{\Gamma\ X^{\|}\ \Delta \Rightarrow Y}\,[{}^{\|}\text{Wkn}]$ $\qquad$ $\dfrac{\Gamma\ X\ \Delta \Rightarrow Y}{\Gamma\ X^{\|}\ \Delta \Rightarrow Y}\,[{}^{\|}\text{L}]$

Again, the structural operation performed by $^{\|}\text{Wkn}$ is that of weakening.

This operator would be used in the type of a lexical item whose argument optionally contained a gap (perhaps as in languages with resumptive pronouns or optional clitic doubling). Such constructions are rare in English, but may occur when an adjective is modified by "too": contrast "the lecture was too boring for me to follow" with "the lecture was too boring for me to stay awake". Let predicative adjectives have the type Pred, and let "for ... to" clauses have the type SP. Then we may capture the above data by giving "too" a type such that "too boring" receives the type $\text{Pred}/(\text{S}/\text{NP}^{\|})$:

(56) 

(57)


The existential optionality modality in $X^?$ indicates zero or one items of type X. The $^?$I rule introduces the operator on a non-modal type as with other existential modalities; the $^?$Stn ('strengthening') rule creates a modal type from nothing:

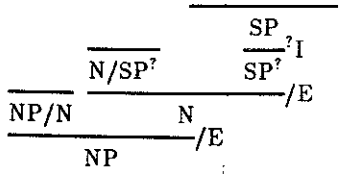
(58) a. $\dfrac{\vdots\ X}{X^?}{}^?\mathrm{I}$ b. $\dfrac{}{X^?}{}^?\mathrm{Stn}$

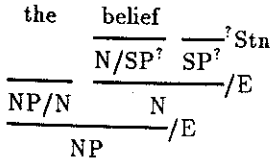
Expressed as sequent proof rules these are as in (59):

(59) $\dfrac{\Gamma \Rightarrow X^?}{\Gamma \Rightarrow X}\ [{}^?\mathrm{R}]$ $\qquad$ $\dfrac{\Gamma\ \Delta\ \Rightarrow Y}{\Gamma\ X^?\ \Delta\ \Rightarrow Y}\ [{}^?\mathrm{Stn}]$

Once again the $^?$Stn rule is associated with an operation which is the inverse of weakening. As with the $^!$Con rule, the premise in the sequent version is more complex than the conclusion, which may cause problems for proof search.

By way of example, the optionality of the sentential complement of *belief* may be characterized by assignment to $\mathrm{N}/\mathrm{SP}^?$:

(60)


(61)


5 Conclusion

This paper has introduced a scheme of natural deduction-like proof figures for the Lambek calculus, and has proposed structural modalities which are suitable for the capture of linguistic generalizations in categorial grammar. It remains to refine the semantics and proof theory, and the development of strategies for proof search offers a particular challenge. For the

present, we hope that the proposals made can be seen as gaining linguistic practicality in the categorial description of natural language, without losing mathematical elegance.

References

Ades, A.E. and Steedman, M.J. (1982). On the order of words. *Linguistics and Philosophy 4*, 517–558.

Ajdukiewicz, K. (1935). Die syntaktische Konnexität. *Studia Philosophica 1*, 1–27. Translated as 'Syntactic connexion' in S. McCall (Ed., 1967), *Polish Logic: 1920–1939*, Oxford University Press, Oxford, 207–231.

Anderson, A.R. and Belnap, N.D. (1975). *Entailment, Volume 1.* Princeton University Press, Princeton.

Bach, E. (1984). Some generalizations of categorial grammars. In F. Landman and F. Veltman (Eds), *Varieties of Formal Semantics*, Foris, Dordrecht.

Bar-Hillel, Y. (1953). A quasi-arithmetical notation for syntactic description. *Language 29*, 47–58.

van Benthem, J. (1983). *The semantics of variety in categorial grammar.* Report 83–29, Department of Mathematics, Simon Fraser University. Also in In W. Buszkowski, W. Marciszewski and J. van Benthem (Eds), *Categorial Grammar*, Volume 25, Linguistic & Literary Studies in Eastern Europe, John Benjamins, Amsterdam/Philadelphia, 57–84.

van Benthem, J. (1986). Categorial grammar. In *Essays in Logical Semantics*, Volume 8, Studies in Linguistics and Philosophy, D. Reidel, Dordrecht, 123–150.

van Benthem, J. (1987). Categorial grammar and type theory. Prepublication Series 87–07, Institute for Language, Logic and Information, University of Amsterdam.

Curry, H.B. and Feys, R. (1958). *Combinatory Logic, Volume I.* North-Holland, Amsterdam.

Gentzen, G. (1936). On the meanings of the logical constants. In Szabo (Ed., 1969), *The Collected Papers of Gerhard Gentzen*, North Holland, Amsterdam.

Girard, J-Y. (1987). Linear logic. *Theoretical Computer Science 50*, 1–102.

Girard, J-Y. (1989). Towards a geometry of interaction. *Proceedings of the AMS Conference on Categories, Logic and Computer Science.*

Hepple, M. (1990). Normal form theorem proving for the Lambek calculus. To appear in *Proceedings of COLING 1990.*

Hepple, M. and Morrill, G. (1989). Parsing and derivational equivalence. In *Proceedings of the Fourth Conference of the European Chapter of the Association for Computational Linguistics*, UMIST, Manchester.

Howard, W.A. (1969). The formulae-as-types notion of construction. In J.R. Hindley and J.P. Seldin (Eds, 1980), *To H.B. Curry, Essays on Combinatory Logic, Lambda Calculus and Formalism*, Academic Press.

König, E. (1989). Parsing as natural deduction. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics.*

Lambek, J. (1958). The mathematics of sentence structure. *American Mathematical Monthly 65*, 154–170.

Moortgat, M. (1988). *Categorial Investigations: Logical and Linguistic Aspects of the Lambek Calculus*, Foris, Dordrecht.

Moortgat, M. (1990a). Cut elimination and the elimination of spurious ambiguity. In *Proceedings of the Seventh Amsterdam Colloquium*, University of Amsterdam.

Moortgat, M. (1990b). The logic of discontinuous type constructors. To appear in *Proceedings of the Symposium on Discontinuous Constituency*, Institute for Language Technology and Information, University of Tilburg.

Morrill, G. (1989). Intensionality and boundedness. To appear in *Linguistics and Philosophy.*

Ono, H. (1988). Structural rules and a logical hierarchy. To appear in *Proceedings of the Conference on Mathematical Logic and its Applications*, North-Holland, Amsterdam.

Prawitz, D. (1965). *Natural Deduction: a Proof Theoretical Study.* Almqvist and Wiksell, Uppsala.

Wansing, H. (1989). Relevant quasi-deductions, weak implicational logics, and operational semantics. Ms., Institut für Philosophie, Freie Universität Berlin.