Grammar and Logical Types*

Glyn Morrill

Centre for Cognitive Science, University of Edinburgh
2 Buccleuch Place, Edinburgh EH8 9LW Scotland

Abstract

This paper represents categorial grammar as an implicational type theory in the spirit of Girard's linear logic, and illustrates linguistic applications of a range of type-constructors over and above implication. The type theoretic perspective is concerned with a correspondence between the logic of types, and computational operations over the objects inhabiting types. In linguistic applications this correspondence is between rules of grammar which are theorems of type inference, and compositional operations in the various algebras in which linguistic objects, i.e. signs, are assumed to have dimensions: syntax, semantics, etc. Rule-to-rule description is familiar from Montague Grammar, but the idea here is to classify signs with structured types satisfying universal type laws determined by the semantics of the type connectives, in contrast to classification by categories satisfying stipulated rules. On this scheme an object language is to be specified by a type assignment to its finite vocabulary: a formal grammar is just a lexicon, plus perhaps some improper type axioms, and a grammar formalism is just a meta-language of types with its uniform logic and interpretation is each linguistic dimension. The aim is to develop a language of types which has sufficient transparency, sensitivity, and generality to implement interesting descriptions of natural language. The paper will illustrate sentence grammar, and also use of the semantic term algebra as a functional programming language for presentation of lexical semantics.

1 Introduction

The grammatical architecture exemplified in Montague Grammar is one which sees linguistic objects as having dimensions in syntactic and semantic algebraic domains, and in which rules of grammar correlate operations in these algebras. For this design in general, a linguistic object or sign is a vector across the elements in the linguistic domains under consideration, and a rule is a vector across the operations; a language is described by the closure of the rules of grammar over the lexical signs.[1]

This paper develops a tool for language description by extending the categorial type system of Lambek (1958).[2] Syntactic interpretation of Lambek categorial types is reviewed in e.g. Buszkowski (1988), and semantic interpretation in van Benthem (1983). On the present design these schemes are to be integrated in a compositional interpretation for type-constructors specifying the sign vectors in the composite types in terms of the sign vectors in the operand types. The rules of grammar — theorems of type inference — are

*The development of this material over a period of time has benefited from a wide range of influences; for extensive comments and suggestions I thank most of all Guy Barry, Mark Hepple, and Neil Leslie.

[1] For discussion of the rule-to-rule grammatical architecture see Oehrle (1988).

[2] See Moortgat (1989).

the validities according to the interpretation of types. The extension of Lambek categorial grammar given is related to linear logic (see e.g. Girard 1989) which drops structural rules from intuitionistic logic; in the present application this corresponds to the fact that grammar is concerned with e.g. presence, and number of occurrnces, of linguistic objects.

Categorial grammar implements in a rather direct way a mode of linguistic analysis which is due in essence to Frege. Under such analysis certain not necessarily basic expressions are taken to be the primary bearers of meaning, and other expressions are attributed with meanings in terms of the meanings of the expressions in which they occur. Bidirectional categorial grammar provides for the classification of linguistic objects by starting with primitive types representing 'complete' (or: primarily meaningful) expressions, and building further classes by means of type-constructors / and \. Linguistic objects will be taken to have two dimensions, syntax and semantics. Assignment of a type X/Y ($Y\backslash X$) to an expression can be seen as a simultaneous classification according to form and meaning stating that the expression prefixes (postfixes) itself to expressions of type Y to form expressions of type X, and stating that the meaning of the resultant expression is given by the application of the meaning of the affix expression to that of the stem expression.

Consider for instance a language containing as complete expressions some proper names "John", "Mary", ...indexed by type NP and some sentences "John walks", "Mary walks", ..., "John likes John", ... indexed by type S. Then "walks" has type NP\S; so also do "likes John", "likes Mary", ..., and "likes" has type (NP\S)/NP. But other type assignments are valid under the intended meaning of the type-constructors. The expression "likes" also has type NP\(S/NP); "John" also has types S/(NP\S), ((NP\S)/NP)\(NP\S), and so on.

As a basis for grammar, the idea is to assign types to the vocabulary in such a way that the rest of the language, and the inhabitation of the sentence type in particular, is determined by rules of type inference. Assume for instance additional types PP and N for prepositional phrases and common noun phrases. Then types may be assigned to words as follows:

(1)

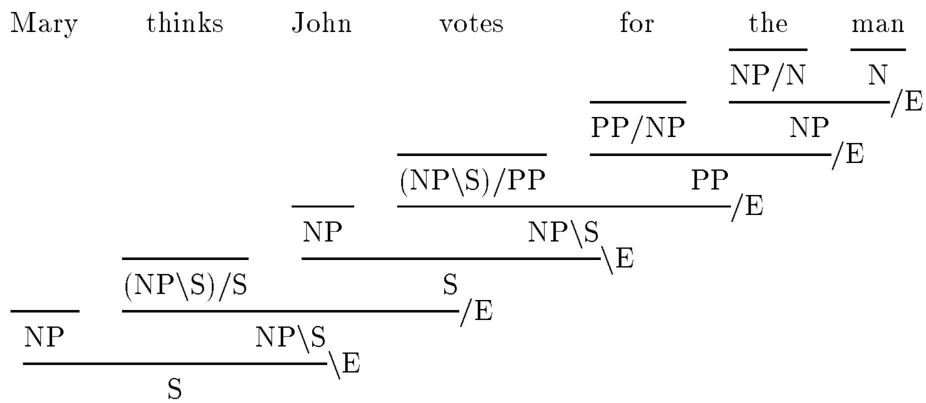
for	:= PP/NP
John, Mary	:= NP
likes	:= (NP\S)/NP
man	:= N
the	:= NP/N
thinks	:= (NP\S)/S
votes	:= (NP\S)/PP

The categorial calculus AB, essentially that of Ajdukiewicz (1935) and Bar-Hillel (1953), contains just the following two rules which state that (*functor*) expressions of types X/Y and $Y\backslash X$ combined with (*argument*) expressions of types Y form expressions of types X.

(2) a.
$$\frac{\overset{\vdots}{X/Y} \qquad \overset{\vdots}{Y}}{X}/\mathrm{E}$$
b.
$$\frac{\overset{\vdots}{Y} \qquad \overset{\vdots}{Y\backslash X}}{X}\backslash\mathrm{E}$$

A type under vertical ellipses signifies a derivation of that type, and a sole type constitutes a derivation of itself. By way of example, "Mary thinks John votes for the man" is derived as a sentence as follows:

(3)



The 'ordered natural deduction' representation of categorial derivations used here is presented in Morrill, Leslie, Hepple, and Barry (1990). It has the convenient property that deductions are syntactically interpreted as the left-to-right concatenation of the premiss forms. Thus the rules of deduction in (2) are syntactically interpreted as concatenation of the subexpressions in left-to-right order, and semantically interpreted as *functional application* of the functor to the argument. The semantics of the composite expression in (3) is as follows, where the semantic operation of application is written as ' and elements of the semantic algebra are represented in boldface.

(4) (**thinks** ' ((**votes** ' (**for** ' (**the** ' **man**))) ' **John**)) ' **Mary**

Not all rules of inference are expressible in the inference figure format being used, and sequent rules will also be given. A sequent is of the form $\Gamma \Rightarrow X$ where Γ is a sequence of types, and asserts that there is a deduction of the *consequent* type X from the *antecedent* types Γ, i.e. that inhabitation of the antecedent types implies inhabitation of the consequent type. Sequents can be annotated with operators in the syntactic and semantic algebras indicating how the syntax and semantics of a consequent type inhabitant can be constructed out of the syntactic and semantic components of signs inhabiting the antecedent types. However, since the syntactic constructions here are always concatenation in left-to-right order, annotation will be limited to semantics.

(5)

$$x : X \Rightarrow x : X \quad [\text{Ax}]$$

$$\begin{array}{l} \Delta(\Gamma) \Rightarrow y : Y \quad [\text{CUT}] \\ \qquad \Gamma \Rightarrow x : X \\ \qquad \Delta(x : X) \Rightarrow y : Y \end{array}$$

$$\begin{array}{l} \Gamma(w : X/Y\ \Delta) \Rightarrow z : Z \quad [/\text{L}] \\ \qquad \Delta \Rightarrow y : Y \\ \qquad \Gamma(w\text{'}y : X) \Rightarrow z : Z \end{array}$$

$$\begin{array}{l} \Gamma(\Delta\ w : Y\backslash X) \Rightarrow z : Z \quad [\backslash\text{L}] \\ \qquad \Delta \Rightarrow y : Y \\ \qquad \Gamma(w\text{'}y : X) \Rightarrow z : Z \end{array}$$

The calculus AB does not capture all the type inferences that are valid according to the interpretation of types described above, e.g. it does not capture that inhabitation of type NP by "Mary" together with inhabitation of type (NP\S)/NP by "likes" implies inhabitation of type S/NP (by "Mary likes"). This completeness is achieved in the associative Lambek calculus L (Lambek 1958). L is obtained in the present 'natural deduction' format by adding the following conditionalisation rules to the directional modus ponens rules of AB. These state that from a derivation of Y from assumptions including an (appropriate) occurrence of X, a further derivation is obtained by withdrawing that assumption occurrence.

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

There is the condition that X is respectively the rightmost and leftmost undischarged assumption above Y in /I and \I; there is also a condition that a sole assumption in a derivation cannot be withdrawn; thus no types are assigned to the empty string. That exactly one assumption must be withdrawn makes this an *occurrence* logic like linear logic; the directional system is non-commutative linear logic. In the introduction rules, the meaning of the result is given by the *functional abstraction* over the meaning of the discharged assumption. The sequent rules are as follows:

(7) $\dfrac{\Gamma \Rightarrow \lambda xy : Y/X \quad [/\mathrm{R}]}{\Gamma \quad x : X \Rightarrow y : Y}$ $\qquad$ $\dfrac{\Gamma \Rightarrow \lambda xy : X\backslash Y \quad [\backslash\mathrm{R}]}{x : X \quad \Gamma \Rightarrow y : Y}$

The rules in (6) are analogous to the usual natural deduction rule of $\rightarrow$I, except exactly one assumption, in a particular place, must be withdrawn. Derivations are semantically interpreted in a single-bind lambda calculus satisfying the usual law of β-reduction (unary operators such as λx will be assumed to bind more tightly than binary operators such as '):

(8) $\lambda x\alpha \text{ ' } \beta = \alpha[\beta/x]$

This Curry-Howard 'formulas-as-types' interpretation of categorial deductions is introduced in van Benthem (1983) and implemented in a sequent context in Moortgat (1989). The simplification in (8) means that introducing and then eliminating implication constitutes a detour in derivation, producing a reducible, non-normal form, proof (cf. Prawitz 1965).

By way of illustration of conditionalisation, the non-canonical constituent "John likes" can be derived as of type S/NP, and assignment of type (N\N)/(S/NP) to an object relative pronoun produces relativisation: (cf. Ades and Steedman 1982).

(9)

$$\dfrac{\dfrac{\text{the}}{\mathrm{NP/N}} \quad \dfrac{\dfrac{\text{man}}{\mathrm{N}} \quad \dfrac{\dfrac{\text{who}}{\mathrm{(N\backslash N)/(S/NP)}} \quad \dfrac{\dfrac{\dfrac{\text{Mary}}{\mathrm{NP}} \quad \dfrac{\dfrac{\text{likes}}{\mathrm{(NP\backslash S)/NP}} \quad [\mathrm{NP}]^1}{\mathrm{NP\backslash S}}/\mathrm{E}}{\mathrm{S}}\backslash\mathrm{E}}{\mathrm{S/NP}}/\mathrm{I}^1}{\mathrm{N\backslash N}}/\mathrm{E}}{\mathrm{N}}\backslash\mathrm{E}}{\mathrm{NP}}/\mathrm{E}$$

The construction for the semantics is in (10).

(10) **the** ' ((**who** ' λx((**likes** ' x) ' **Mary**)) ' **man**)

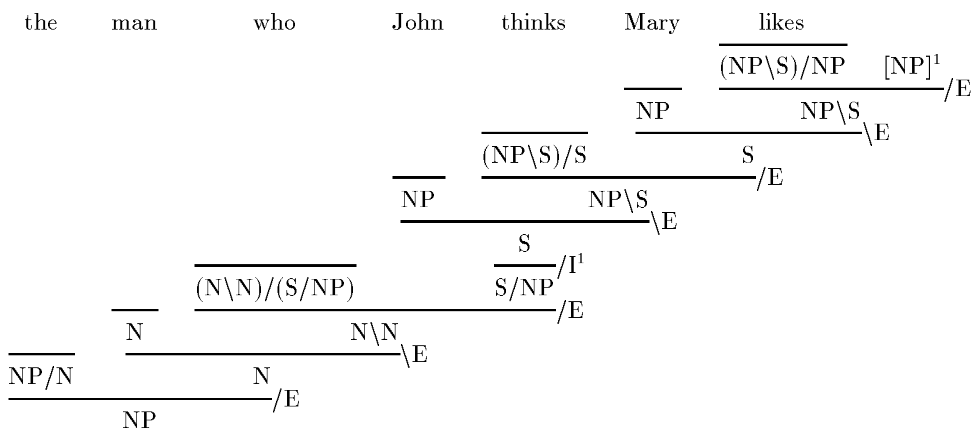
This specifies the meaning of the composite expression in terms of the meanings of the words. In order to give content to the semantics of words, these can themselves be expressed as terms in the semantic algebra. Thus the meaning of a relative pronoun might be described as follows:

(11) $\mathbf{who} = \lambda x \lambda y \lambda z((\mathbf{AND}\ '\ (x\ '\ z))\ '\ (y\ '\ z))$

Then (10) simplifies as shown in (nextex) (throughout variables will be renamed as is convenient).

(12) $\mathbf{the}\ '\ ((\lambda x \lambda y \lambda z((\mathbf{AND}\ '\ (x\ '\ z))\ '\ (y\ '\ z))\ '\ \lambda w((\mathbf{likes}\ '\ w)\ '\ \mathbf{Mary}))\ '\ \mathbf{man}) =$
$\mathbf{the}\ '\ (\lambda y \lambda z((\mathbf{AND}\ '\ (\lambda w((\mathbf{likes}\ '\ w)\ '\ \mathbf{Mary})\ '\ z))\ '\ (y\ '\ z))\ '\ \mathbf{man}) =$
$\mathbf{the}\ '\ \lambda z((\mathbf{AND}\ '\ ((\mathbf{likes}\ '\ z)\ '\ \mathbf{Mary}))\ '\ (\mathbf{man}\ '\ z))$

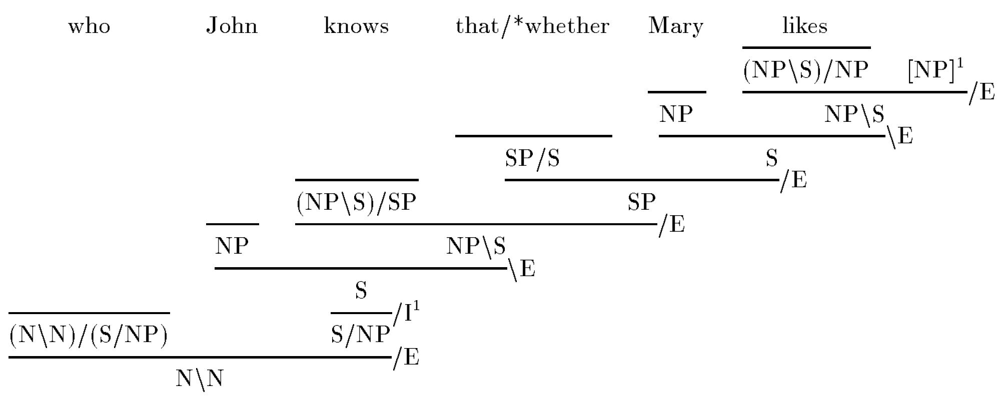
The operations ' and λx are like APPLY and LAMBDA (X) in the functional programming language Lisp, which was partly inspired by the lambda calculus. This treatment allows for long-distance relativisation:

(13)


The meaning representation in (14) would reduce much as in (12).

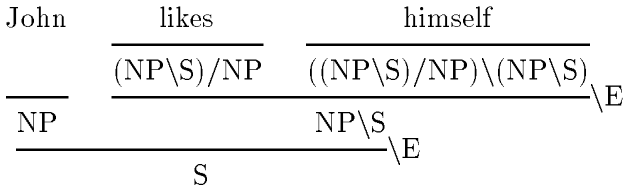
(14) $\mathbf{the}\ '\ ((\mathbf{who}\ '\ \lambda x((\mathbf{thinks}\ '\ ((\mathbf{likes}\ '\ x)\ '\ \mathbf{Mary}))\ '\ \mathbf{John}))\ '\ \mathbf{man})$

However, a straightforward type assignment does not capture the following contrast (an instance of Chomsky's *wh*-island constraint):

(15)


In relation to another kind of discontinuity in natural language, Szabolcsi (1987) observes that assignment of type ((NP\S)/NP)\(NP\S) to reflexives provides a characterisation of reflexivisation:

(16)

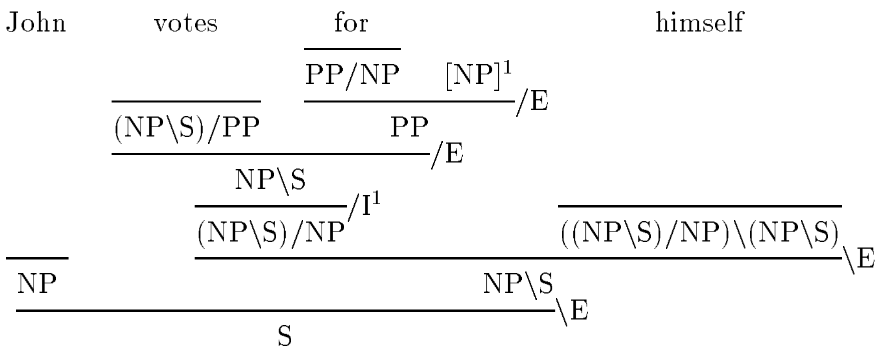


She notes that the reflexive should have the semantics of combinator **W** (lambda term $\lambda x \lambda y((x \text{ ' } y) \text{ ' } y)$). Desclés, Guentchéva and Shaumyan (1986) also make this proposal.

(17) $(\lambda x \lambda y((x \text{ ' } y) \text{ ' } y) \text{ ' } \mathbf{likes}) \text{ ' } \mathbf{John} =$
$\lambda y((\mathbf{likes} \text{ ' } y) \text{ ' } y) \text{ ' } \mathbf{Mary} =$
$(\mathbf{likes}'\mathbf{John})'\mathbf{John}$

Conditionalisation allows for less trivial cases:

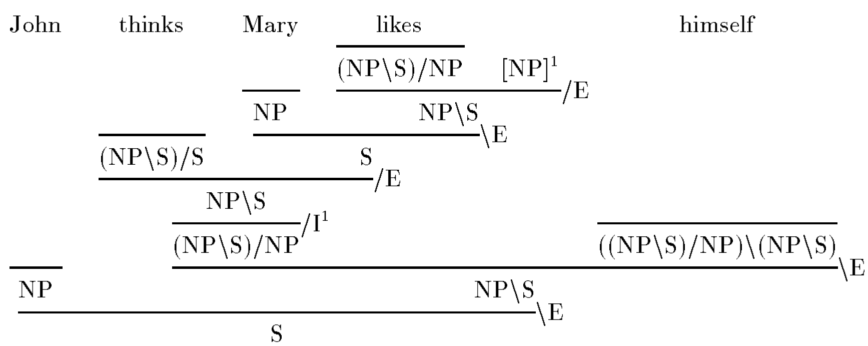
(18)



(19) $(\lambda x \lambda y((x \text{ ' } y) \text{ ' } y) \text{ ' } \lambda z(\mathbf{votes} \text{ ' } (\mathbf{for} \text{ ' } x))) \text{ ' } \mathbf{John} =$
$(\mathbf{votes} \text{ ' } (\mathbf{for} \text{ ' } \mathbf{John})) \text{ ' } \mathbf{John}$

However the current grammar allows a reflexive to take an antecedent from a superordinate clause:

(20) *



(21) $(\lambda x \lambda y((x \text{ ' } y) \text{ ' } y) \text{ ' } \lambda z(\mathbf{thinks} \text{ ' } ((\mathbf{likes} \text{ ' } z) \text{ ' } \mathbf{Mary}))) \text{ ' } \mathbf{John} =$
$(\mathbf{thinks} \text{ ' } ((\mathbf{likes} \text{ ' } \mathbf{John}) \text{ ' } \mathbf{Mary})) \text{ ' } \mathbf{John}$

Such lack of sensitivity towards islandhood in relativisation and boundedness in reflexivisation is symptomatic of the apparent inadequacy of Lambek categorial type theory as a grammar formalism for natural language. There are many other shortcomings. For instance, natural language abounds with ambiguity: homonymy, and also ambiguity with respect to some but not all of the context of an element, e.g. transitive prepositions always take an object, but may then be either adnominal or adverbial, and many verbs can take different complements while uniformly forming verb phrases. Treating all such cases as lexical ambiguity misses the generalisations.

Further problems arise with respect to the kind of polymorphism that is treated with features in unification grammars: introduction of distinct primitive types for subclasses differing in minor features will lose those generalisations that can be made on the basis of major classes.

There are additionally well-known puzzles raised by an assumption of 'like-type' coordination: what type is shared by the conjuncts in "John is rich and an excellent cook"? And in "John or Mary walks" an attempt to match the gender of the conjuncts will fail.

Finally, the Lambek calculus is a sequence logic, and while this is appropriate for classification of linear linguistic forms, it of itself does not offer any control over the subtleties of occurrence and order in natural language.

The thesis of this paper is that with appropriate new type-constructors, categorial grammar can approach the kind of simultaneous sensitivity and generality required for description of natural language. Developing Morrill (1989b) more fully, the paper will introduce successive connectives together with their logic and informal interpretations, applying these to linguistic problems such as those mentioned above.

The calculus of Lambek (1958) actually included already the *product* type-constructor. The type $X \cdot Y$ is interpreted as the set of pairings of objects of types X and Y; there are the following rules of deduction:[3]

(22)
$$\frac{\begin{matrix}\vdots & & \vdots\\ X & & Y\end{matrix}}{X \cdot Y}\cdot\mathrm{I} \qquad \frac{\begin{matrix}\vdots\\ X \cdot Y\end{matrix}}{X \quad\quad Y}\cdot\mathrm{E}$$

The annotated sequent rules are as follows.

(23)
$$\begin{array}{ll} \Gamma \quad \Delta \Rightarrow x, y : X \cdot Y \quad [\cdot\mathrm{R}] & \qquad \Gamma(z : X \cdot Y) \Rightarrow C \quad [\cdot\mathrm{L}] \\ \qquad \Gamma \Rightarrow x : X & \qquad\qquad \Gamma(\pi^1 z : X \quad \pi^2 z : Y) \Rightarrow C \\ \qquad \Delta \Rightarrow y : Y & \end{array}$$

That pairing and projection are appropriate to the interpretation of product is noted in van Benthem (1987); they satisfy the following, being analogous to CONS, CAR and CDR in Lisp.

(24) $\pi_1(\alpha, \beta) = \alpha \qquad \pi_2(\alpha, \beta) = \beta$

These equalities again mean that feeding a product introduction into elimination results in a *redex*, i.e. a proof form which can be simplified. Complementarity in the semantic operations interpreting proofs will mean that the subsequent connectives have this introduction/elimination cancellation, which is a characteristic feature of type theory and its use in functional programming (see e.g. Girard, Taylor, and Lafont 1989). Hobbs and Rosenschein (1978) point out the close relations between Lisp and Montague's IL.

Linguistic applications of product are discussed in Wood (1988). In the following example it is used to give a Government-Binding style 'small-clause' analysis where the object and predicative following the verb form a constituent:

[3] In fact unless product pairing is interpreted by a non-associative concatenation it is not clear that product elimination can be constructively interpreted in the syntax, since the appropriate partitioning cannot be recovered from a string

(25)

$$\dfrac{\dfrac{\text{Mary}}{\text{NP}} \qquad \dfrac{\dfrac{\text{considers}}{(\text{NP}\backslash\text{S})/(\text{NP}\cdot(\text{N}/\text{N}))} \qquad \dfrac{\dfrac{\text{John}}{\text{NP}} \quad \dfrac{\text{tall}}{\text{N}/\text{N}}}{\text{NP}\cdot(\text{N}/\text{N})}\cdot\text{I}}{\text{NP}\backslash\text{S}}/\text{E}}{\text{S}}\backslash\text{E}$$

The semantic construction is given in (26).

(26) $(\mathbf{considers}\ '\ (\mathbf{John}\ ,\ \mathbf{tall}))\ '\ \mathbf{Mary}$

The meaning of "considers" can be further specified:

(27) $\mathbf{considers} = \lambda x \lambda y((\mathbf{CONSIDERS}\ '\ y)\ '\ (\pi^2 x\ '\ \pi^1 x))$

Then (26) reduces thus:

(28) $(\lambda x \lambda y((\mathbf{CONSIDERS}\ '\ y)\ '\ (\pi^2 x\ '\ \pi^1 x))\ '\ (\mathbf{John}\ ,\ \mathbf{tall}))\ '\ \mathbf{Mary} =$
$\lambda y((\mathbf{CONSIDERS}\ '\ y)\ '\ (\pi^2(\mathbf{John}\ ,\ \mathbf{tall})\ '\ \pi^1(\mathbf{John}\ ,\ \mathbf{tall})))\ '\ \mathbf{Mary} =$
$(\mathbf{CONSIDERS}\ '\ \mathbf{Mary})\ '\ (\mathbf{tall}\ '\ \mathbf{John})$

2 Booleans

The conjunction type-constructor $\wedge$ is interpreted syntactically as intersection and semantically as pairing. Thus $X \wedge Y$ is the set of all sign vectors with an ordered pair semantics such that the syntax with the first meaning is a sign in X and the syntax with the second meaning is a sign in Y. There are the following rules of deduction.

(29) a.
$$\dfrac{\begin{matrix}\vdots\\ X \wedge Y\end{matrix}}{X}\wedge\text{E}_a$$
b.
$$\dfrac{\begin{matrix}\vdots\\ X \wedge Y\end{matrix}}{Y}\wedge\text{E}_b$$

The $\wedge$I rule cannot be stated in the ordered natural deduction format but the sequent rules are as follows:

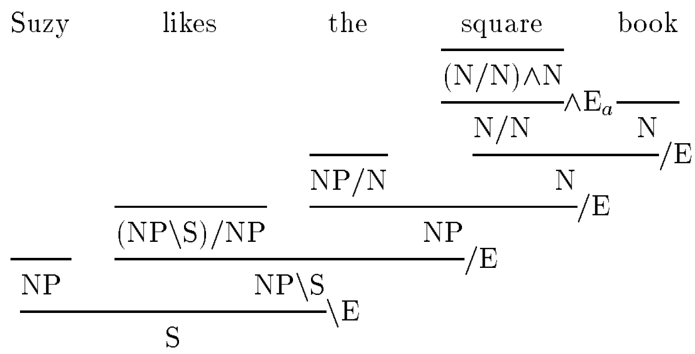
(30)
$$\begin{matrix}\Gamma(z : X \wedge Y) \Rightarrow C \quad [\wedge\text{L}_a]\\ \Gamma(z_1 : X) \Rightarrow C\end{matrix} \qquad\qquad \begin{matrix}\Gamma(z : X \wedge Y) \Rightarrow C \quad [\wedge\text{L}_b]\\ \Gamma(z_2 : Y) \Rightarrow C\end{matrix}$$

(31)
$$\begin{matrix}\Gamma \Rightarrow x * y : X \wedge Y \quad [\wedge R]\\ \Gamma \Rightarrow x : X\\ \Gamma \Rightarrow y : Y\end{matrix}$$

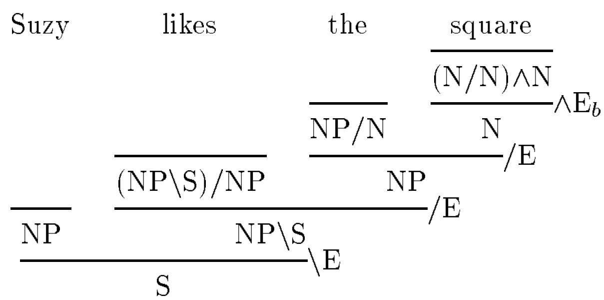
Intuitively the operator $*$ represents a point of nondeterminism and the operators $_1$ and $_2$ decisions to take the first and second branches. They obey the usual laws of pairing and projection:

(32) $(\alpha * \beta)_1 = \alpha \qquad (\alpha * \beta)_2 = \beta$

Then e.g. the homonymy of "square" may now be captured by assignment of the type (N/N)$\wedge$N:

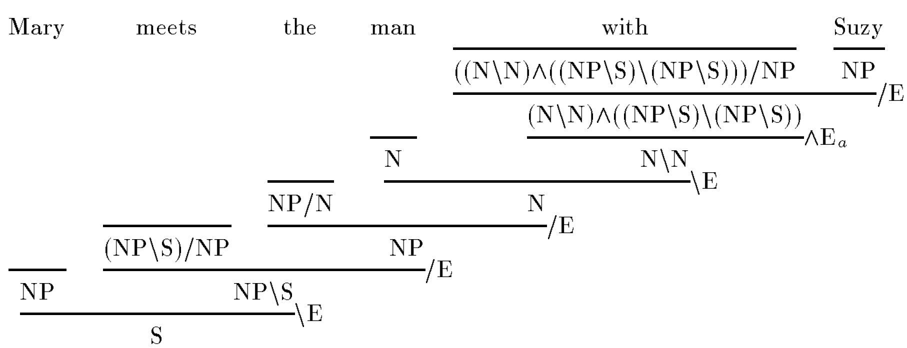
(33) 

(34) (**likes** ’ (**the** ’ (**square**$_1$ ’ **book**))) ’ **Suzy**

(35) 

(36) (**likes** ’ (**the** ’ **square**$_2$)) ’ **Suzy**

The conjunction connective also allows for the description of ‘value ambiguity’, such as that of a preposition which having once combined with its object may play the role of either an adnominal or adverbial. The two readings of “Mary meets the man with Suzy” are obtained from a single type assignment:

(37) 

(38) (**meets** ’ (**the** ’ ((**with** ’ **Suzy**)$_1$ ’ **man**))) ’ **Mary**

(39)
$$\dfrac{\dfrac{\text{Mary}}{\mathrm{NP}} \quad \dfrac{\dfrac{\dfrac{\text{meets}}{(\mathrm{NP}\backslash \mathrm{S})/\mathrm{NP}} \quad \dfrac{\dfrac{\text{the}}{\mathrm{NP}/\mathrm{N}} \quad \dfrac{\text{man}}{\mathrm{N}}}{\mathrm{NP}}/\mathrm{E}}{\mathrm{NP}\backslash \mathrm{S}}/\mathrm{E} \quad \dfrac{\dfrac{\dfrac{\text{with}}{((\mathrm{N}\backslash \mathrm{N})\wedge((\mathrm{NP}\backslash \mathrm{S})\backslash(\mathrm{NP}\backslash \mathrm{S})))/\mathrm{NP}} \quad \dfrac{\text{Suzy}}{\mathrm{NP}}}{(\mathrm{N}\backslash \mathrm{N})\wedge((\mathrm{NP}\backslash \mathrm{S})\backslash(\mathrm{NP}\backslash \mathrm{S}))}/\mathrm{E}}{(\mathrm{NP}\backslash \mathrm{S})\backslash(\mathrm{NP}\backslash \mathrm{S})}\wedge\mathrm{E}_b}{\mathrm{NP}\backslash \mathrm{S}}\backslash\mathrm{E}}{\mathrm{S}}\backslash\mathrm{E}$$

(40) (((**with** ' **Suzy**)$_2$) ' (**meets** ' (**the** ' **man**))) ' **John**

The disjunction type-constructor $\vee$ is interpreted syntactically as union and semantically as disjoint union. Thus $X \vee Y$ is the type of all sign vectors consisting of a syntax, and semantics flagged according as the syntax and unflagged semantics is in X or Y. There are the following rules of deduction:

(41) a.
$$\dfrac{\begin{array}{c}\vdots\\ X\end{array}}{X \vee Y}\vee\mathrm{I}_a$$
b.
$$\dfrac{\begin{array}{c}\vdots\\ Y\end{array}}{X \vee Y}\vee\mathrm{I}_b$$

This time the $\vee$E rule is hard to state as an ordered natural deduction, but the sequent rules are thus:

(42)
$$\begin{array}{c}\Gamma \Rightarrow ix : X \vee Y \quad [\vee\mathrm{R}_a]\\ \Gamma \Rightarrow x : X\end{array} \qquad\qquad \begin{array}{c}\Gamma \Rightarrow jy : X \vee Y \quad [\vee\mathrm{R}_b]\\ \Gamma \Rightarrow y : Y\end{array}$$

(43)
$$\begin{array}{c}\Gamma(w : X \vee Y) \Rightarrow w \rightarrow x.z^i; y.z^j : Z \quad [\vee\mathrm{L}]\\ \Gamma(x : X) \Rightarrow z^i : Z\\ \Gamma(y : Y) \Rightarrow z_j : Z\end{array}$$

Intuitively the operator in (43) is a case operator keyed on i and j:

(44) $i\alpha \rightarrow x.\gamma^i; y.\gamma^j = \gamma^i[\alpha/x]$
$j\alpha \rightarrow x.\gamma^i; y.\gamma^j = \gamma^j[\alpha/y]$

The disjunction connective finds application in e.g. the ambiguity of "wants" with respect to its complementation:

(45)
$$\dfrac{\dfrac{\text{Mary}}{\mathrm{NP}} \quad \dfrac{\dfrac{\text{wants}}{(\mathrm{NP}\backslash \mathrm{S})/(\mathrm{VP}\vee(\mathrm{NP}\cdot\mathrm{VP}))} \quad \dfrac{\dfrac{\text{to-go}}{\mathrm{VP}}}{\mathrm{VP}\vee(\mathrm{NP}\cdot\mathrm{VP})}\vee\mathrm{I}_a}{\mathrm{NP}\backslash \mathrm{S}}/\mathrm{E}}{\mathrm{S}}$$

(46) (**wants** ' i**to-go**) ' **Mary**

(47) Mary wants John to-go

$$\dfrac{\dfrac{}{\mathrm{NP}} \qquad \dfrac{\dfrac{}{(\mathrm{NP}\backslash \mathrm{S})/(\mathrm{VP}\vee(\mathrm{NP}\cdot \mathrm{VP}))} \qquad \dfrac{\dfrac{\dfrac{\text{John}}{\mathrm{NP}}\quad\dfrac{\text{to-go}}{\mathrm{VP}}}{\mathrm{NP}\cdot \mathrm{VP}}\,\cdot \mathrm{I}}{\mathrm{VP}\vee(\mathrm{NP}\cdot \mathrm{VP})}\,\vee \mathrm{I}_b}{\mathrm{NP}\backslash \mathrm{S}}\,/\mathrm{E}}{\mathrm{S}}\,\backslash \mathrm{E}$$

(48) (**wants** ' j(**John** , **to-go**)) ' **Mary**

The semantics of **wants** can be spelled out in terms of a more primitive **WANTS** as follows:

(49) $\mathbf{wants} = \lambda x\lambda y((\mathbf{WANTS}\ '\ y)\ '\ x \to z.(z\ '\ y); w.(\pi^2 w\ '\ \pi^1 w))$

Then (46) simplifies as in (50).

(50) $(\lambda x\lambda y((\mathbf{WANTS}\ '\ y)\ '\ x \to z.(z\ '\ yx); w.(\pi^2 w\ '\ \pi^1 w))\ '\ i\mathbf{to\text{-}go})\ '\ \mathbf{Mary} =$
$(\mathbf{WANTS}\ '\ \mathbf{Mary})\ '\ i\mathbf{to\text{-}go} \to z; (z\ '\ \mathbf{Mary}); w.(\pi^2 w\ '\ \pi^1 w) =$
$(\mathbf{WANTS}\ '\ \mathbf{Mary})\ '\ (\mathbf{to\text{-}go}\ '\ \mathbf{Mary})$

Similarly, (48) reduces as in (51).

(51) $(\lambda x\lambda y((\mathbf{WANTS}\ '\ y)\ '\ x \to z.(z\ '\ yx); w.(\pi^2 w\ '\ \pi^1 w))\ '\ j(\mathbf{John}\ ,\ \mathbf{to\text{-}go}))\ '$
$\mathbf{Mary} =$
$(\mathbf{WANTS}\ '\ \mathbf{Mary})\ '\ j(\mathbf{John}\ ,\ \mathbf{to\text{-}go}) \to z.(z\ '\ \mathbf{Mary}); w.(\pi^2 w\ '\ \pi^1 w) =$
$(\mathbf{WANTS}\ '\ \mathbf{Mary})\ '\ (\pi^2(\mathbf{John}\ ,\ \mathbf{to\text{-}go})\ '\ \pi^1(\mathbf{John}\ ,\ \mathbf{to\text{-}go})) =$
$(\mathbf{WANTS}\ '\ \mathbf{Mary})\ '\ (\mathbf{to\text{-}go}\ '\ \mathbf{John})$

Disjunction suggests a means by which coordination of unlike types can be licensed if there is a suitable functor over the disjunction of their domains:

(52) a. John is rich.

b. John is an excellent cook.

c. John is rich and an excellent cook.

(53) is rich an excellent cook

$$\dfrac{\text{is}}{(\mathrm{NP}\backslash \mathrm{S})/(\mathrm{NP}\vee(\mathrm{N}/\mathrm{N}))} \qquad \dfrac{\dfrac{\text{rich}}{\mathrm{N}/\mathrm{N}}}{\mathrm{NP}\vee(\mathrm{N}/\mathrm{N})}\,\vee \mathrm{I}_b \qquad \dfrac{\dfrac{\text{an excellent cook}}{\mathrm{NP}}}{\mathrm{NP}\vee(\mathrm{N}/\mathrm{N})}\,\vee \mathrm{I}_a$$

The conjunction type actually appears in Lambek (1961), and use of booleans is implied in van Benthem (1989a). Keenan and Timberlake (1988) use an 'n-tuple' type constructor such that if $X_1, Y_1, \ldots, X_n, Y_n$ are types then so is $< X_1, \ldots, X_n > / < Y_1, \ldots, Y_n >$. This would be defined in terms of the proposal above as $(X_1/Y_1) \wedge \ldots \wedge (X_n/Y_n)$; the n-tuple types for the earlier "with" and "wants" would be $<$ [N\N, (NP\S)\(NP\S)] $>$ /[NP, NP] $>$ and $<$ [NP\S, NP\S] $>$ /[VP, NP·VP] $>$, i.e. they do not capture the generalisations that domains or ranges are the same.

A negation type-constructor could be syntactically interpreted as set complement. A universal type t and null type $\bot$ have logic as follows:

(54) $\Gamma \Rightarrow \mathrm{t} \qquad\qquad \bot \Rightarrow X$

The product unit 1 is as in (55).

(55) $\Rightarrow 1 \qquad \dfrac{\Gamma(1) \Rightarrow X}{\Gamma() \Rightarrow X}$

3 Quantification

The proposal of this section is to achieve increased sensitivity by moving from a *propositional* system of types, to a *predicational* one. Unification will be represented as being a way of implementing part of such a proposal.

Instead of just the primitive types as propositional non-logical constants, there will now be feature, feature-function, and predicate constants. Thus the type of a feminine noun phrase might be NP(f), or aiming for more information the type of a nominative third person feminine noun phrase might be NP(third(f), nom). In order to ensure coherent occupency of argument positions, the system should be *sorted*, i.e. distinguishing gender from case, etc.

A facility of variables and quantification over features enables description of polymorphisms. A universally quantified type signifies feature-dependent elements which for any feature of the quantified sort can adapt to a member of the class represented by the body of the quantified type under a valuation where the quantified variable is assigned that feature.

The rules of deduction for universal quantification are as follows:

(56) $$\frac{\begin{array}{c}\vdots\\ X\end{array}}{\forall vX}\forall \text{I} \qquad \frac{\begin{array}{c}\vdots\\ \forall vX\end{array}}{X[f/v]}\forall \text{E}$$

There is the condition on ∀I that v is not free in any undischarged assumption above X, and in ∀E f must be a feature with the sort of v. The introduction rule is semantically interpreted as abstraction over the quantified sort, and the elimination as application to features of the quantified sort. The annotated sequent rules are as follows:

(57) $\dfrac{\Gamma \Rightarrow \text{L}vx : \forall vX}{\Gamma \Rightarrow x : X}[\forall\text{R}] \qquad \dfrac{\Gamma(x : \forall vX) \Rightarrow y : Y}{\Gamma(x' f : X[f/v]) \Rightarrow y : Y}[\forall\text{L}]$

The feature application and abstraction operators satisfy the usual lambda conversion; feeding ∀I into ∀E produces a proof redex.

(58) $\text{L}v\alpha\ '\ F = \alpha[F/v]$

Existentially quantified types are semantically interpreted as pairs consisting of a feature of the quantified sort, and a member of the body domain under a valuation where the quantified variable is assigned that feature. The elimination deduction cannot be represented by an ordered natural deduction inference figure, but the introduction rule is thus:

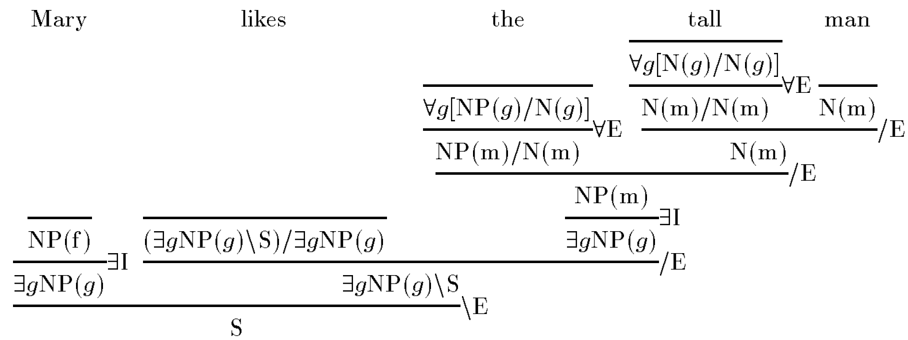
(59) $$\frac{\begin{array}{c}\vdots\\ X[f/v]\end{array}}{\exists vX}\exists \text{I}$$

The sequent rules are as follows:

(60) $\dfrac{\Gamma \Rightarrow f\text{‘}x : \exists vX}{\Gamma \Rightarrow x : X[f/v]}[\exists\text{R}] \qquad \dfrac{\Gamma(x : \exists vX) \Rightarrow Y}{\Gamma(2x : X[1x/v]) \Rightarrow Y}[\exists\text{L}]$

(61) $1(\alpha \text{ ‘ } \beta) = \alpha \qquad 2(\alpha \text{ ‘ } \beta) = \beta$

Type assignments and derivations may now be as illustrated in (62).

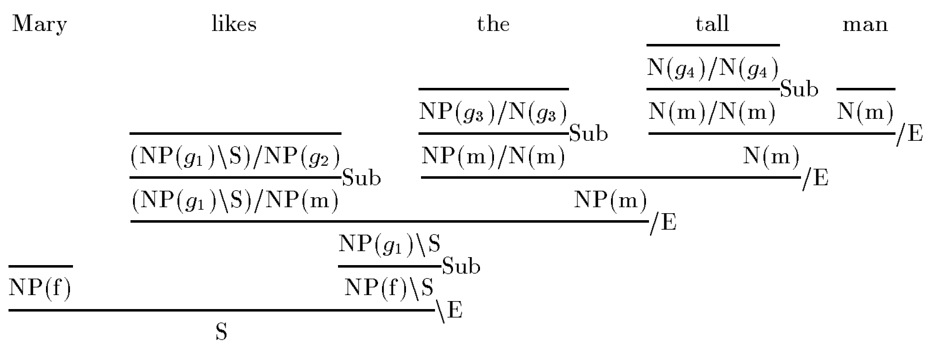
(62)


(63) (**likes** ’ (m ‘ ((**the** ′ m) ’ ((**tall** ′ m) ’ **man**)))) ’ (f ‘ **Mary**)

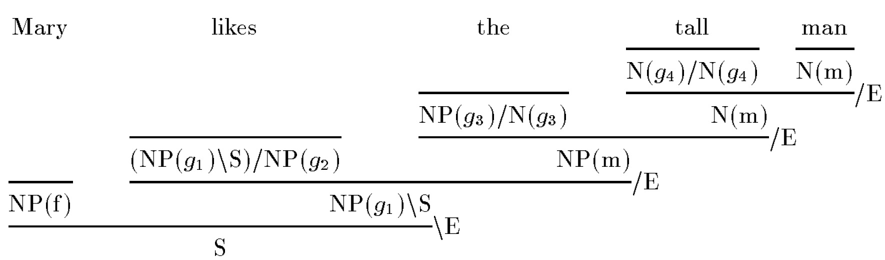
In the case that there is a quantifier free logic with all variables implicitly quantified at the outermost level, this scheme looks like a term unification formalism such as that used in Unification Categorial Grammar (Zeevat, Klein, and Calder 1987); the universal quantifier rules are replaced by a rule of substitution:

(64)
$$\frac{\begin{matrix}\vdots \\ X\end{matrix}}{X[a/v]}\text{Sub}$$

Then the previous derivation may be as follows:

(65)


Alternatively, as noted in van Benthem (1989b), a system of deduction based on resolution (such as that implemented in Prolog) could be used. For AB deductions, a most general unifier of a functor's argument type and an actual argument type is to be applied to the functor's value to obtain the actual value:

(66)


Finally, type structures could be notated as sets of paths, e.g. NP(third(f), nom) could be encoded [[Maj NP] [Min [Agr [[Per third] [Gen f]]] [Case nom]]], resulting in a graph unification formalism such as that used in Head-Driven Phrase Structure Grammar (Pollard and Sag 1987).

In term and graph unification formalisms it is possible to use the implicit universal quantification implemented by unification to assemble semantic (and syntactic) representations within the type structures. With explicit quantification however, the lambda semantics for universally quantified types is seen to be just like that for any other type-constructor, with the usual type theoretic relation between proofs over types, and functional terms. Limitation to universal quantification retains much expressive power, but it may be noted that existential quantification will provide a shared type for the conjuncts in such cases as (67) which universal quantification could not provide without the disjunction of the previous section.

(67) John or Mary walks.

(68)
$$\dfrac{\dfrac{\text{John}}{\text{NP(m)}}}{\exists g \text{NP}(g)}\exists\text{I} \qquad \dfrac{\dfrac{\text{Mary}}{\text{NP(f)}}}{\exists g \text{NP}(g)}\exists\text{I}$$

The current proposals do not include boolean operations on features as in e.g. NP(f&m), though there seems no reason why such logical feature-functions should not be included. Another natural generalisation is to allow second order type quantification — quantification over types rather than just features — in order to describe type dependent elements; van Benthem (1989b) discusses how this can be interpreted in second order lambda calculus.

4 Universal Modality

Morrill (1989a, to appear) introduces a modal operator such that $\Box X$ is the type of intensions of expressions of type X. The S4 rules of deduction are as follows:

(69)
$$\dfrac{\begin{array}{c}\vdots\\ X\end{array}}{\Box X}\Box\text{I} \qquad \dfrac{\begin{array}{c}\vdots\\ \Box X\end{array}}{X}\Box\text{E}$$

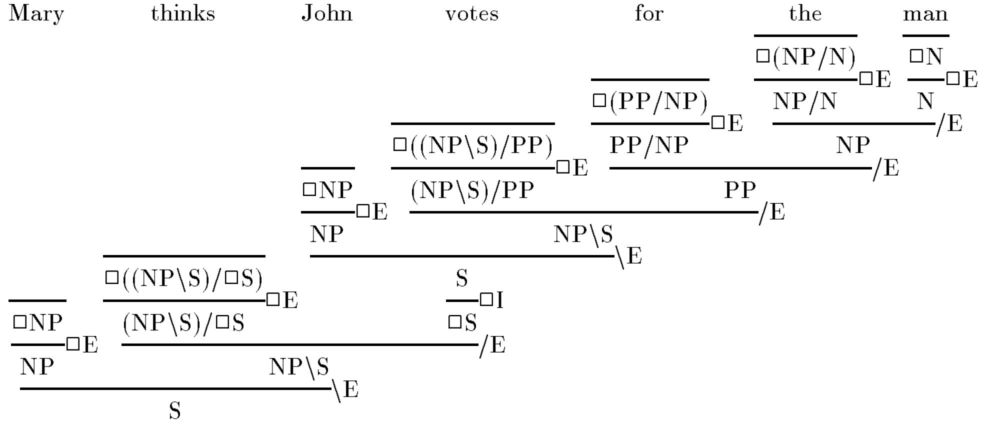
There is the condition on $\Box$I that every path from the root to an undischarged assumption contains a licensing modal type, i.e. a modal type which does not depend on discharged assumptions. The introduction and elimination is semantically interpreted as intensionalisation and extensionalisation. These operators annotate the S4 sequent rules ($\Box\Gamma$ represents a sequence of modal types).

(70)
$$\dfrac{\Box\Gamma \Rightarrow x : X}{\Box\Gamma \Rightarrow \hat{}x : \Box X}[\Box\text{R}] \qquad \dfrac{\Gamma(\check{}y : X) \Rightarrow C}{\Gamma(y : \Box X) \Rightarrow C}[\Box\text{L}]$$

The intension and extension operators satisfy the law of 'down-up cancellation', and feeding $\Box$I into $\Box$E produces a proof redex; these operators have analogues in the QUOTE and EVAL of Lisp.

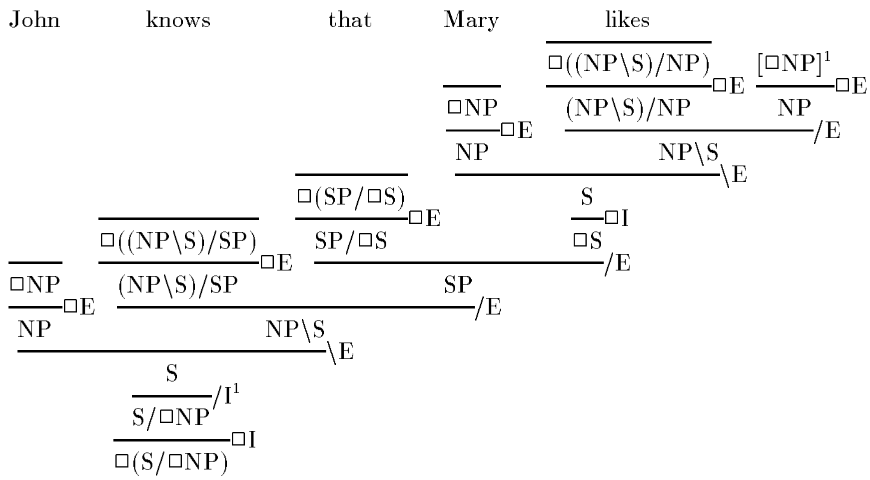
(71) $\check{}\,\hat{}x = x$

An intensional analysis of “Mary thinks John votes for the man” is given in (72).

(72)


(73) (ˇ**thinks** ' ˆ(ˇ**votes** ' (ˇ**for** ' (ˇ**the** ' ˇ**man**))) ' **John**) ' **Mary**

Consider the effect of assigning a relative pronoun type □(N\N)/□(S/□NP). Then the body of “who John knows that Mary likes” is obtained as follows:

(74)


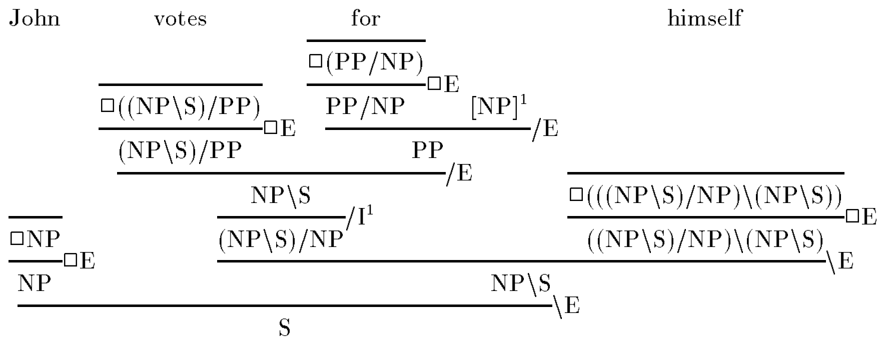
The meaning of the embedded clause is thus:

(75) **who** ' ˆλx((ˇ**knows** ' (ˇ**that** ' ˆ((ˇ**likes** ' ˇx) ' **Mary**)) ' **John**))

However, if “whether” is of type □SP/□S, the earlier *wh*-island violation is not obtained: the sequent in (76) is not a theorem.

(76) $\dfrac{\text{John}}{\Box NP}\ \dfrac{\text{knows}}{\Box((NP\backslash S)/SP)}\ \dfrac{\text{whether}}{\Box SP/\Box S}\ \dfrac{\text{Mary}}{\Box NP}\ \dfrac{\text{likes}}{\Box((NP\backslash S)/NP)} \Rightarrow \Box(S/\Box NP)$

Reflexivisation might be treated as follows.

(77) 

(78) (**himself** ' λx(ˇ**votes** ' (ˇ**for** ' x))) ' **John**

Then the long distance reflexivisation is no longer obtained; the sequent in (79) is not a theorem.

(79) $\frac{\text{Bill}}{\Box\text{NPm}} \quad \frac{\text{thinks}}{\Box((\text{NPm}\backslash\text{S})/\Box\text{S})} \quad \frac{\text{Mary}}{\Box\text{NPf}} \quad \frac{\text{likes}}{\Box((\text{NPf}\backslash\text{S})/\text{NPm})} \quad \frac{\text{himself}}{\Box(((\text{NPm}\backslash\text{S})/\text{NPm})\backslash(\text{NPm}\backslash\text{S}))} \Rightarrow \text{S}$

5 Structural Operations

In an *occurrence* logic such as linear logic, the structural rules of weakening and contraction are not valid: the presence and number of occurrences of a premiss are significant. However, Girard reintroduces these operations via *exponentials* or *structural modalities* which are unary type connectives that essentially license the structural rules on formulas bearing the appropriate connective.

Taking this strategy as inspiration, Morrill, Leslie, Hepple, and Barry (1990) present structural modalities for categorial calculi. Linear logic preserves the structural rule of permutation; since this is also dropped in sequence logics, like the Lambek calculus, we will again adopt Girard's strategy and invoke exchange exponentials. Although the proposals are in the spirit of linear logic, they will be tuned to the sequence calculi and linguistic applications that are the current concern, and similarities in notation do not imply identity with standard linear logic.

5.1 Optionality

The structural rule of weakening is as follows:

(80) $\dfrac{\Gamma\ Y \Rightarrow X}{\Gamma \Rightarrow X}$ [W]

The assertion is that when some premisses yield a conclusion, the conclusion is still yielded when a further premiss is added. The optionality of the presence of a premiss has a linguistic analogue in the optionality of certain elements. Assume a unary connective ? such that $?X$ is the type of those expressions which are either an X, or else are the empty expression. A sequence $\Gamma(?X)$ can then be regarded as abbreviating the disjunction $\Gamma(X)$ or $\Gamma()$. The following rules of deduction are valid.

(81) a. $\dfrac{\begin{array}{c}\vdots\\ X\end{array}}{?X}\,?\mathrm{I}_1$ b. $\dfrac{}{?X}\,?\mathrm{I}_2$

Expressed as sequent proof rules these are as in (82); the rule in (83) is also valid.

(82) a. $\Gamma \Rightarrow ?X \quad [?\mathrm{R}_1]$
$\quad\Gamma \Rightarrow X$

b. $\Rightarrow ?X \quad [?\mathrm{R}_2]$

(83) $\Gamma(?X) \Rightarrow Y \quad [?\mathrm{L}]$
$\quad\Gamma(X) \Rightarrow Y$
$\quad\Gamma() \Rightarrow Y$

Then for example, the optionality of the sentential complement of *belief* is characterised by assignment to N/?SP:

(84) the belief that John lies

$$\dfrac{\dfrac{}{\mathrm{NP/N}} \quad \dfrac{\dfrac{}{\mathrm{N/?SP}} \quad \dfrac{\dfrac{\dfrac{\text{that John lies}}{}}{\mathrm{SP}}}{\mathrm{?SP}}\,?\mathrm{I}_1}{\mathrm{N}}\,/\mathrm{E}}{\mathrm{NP}}\,/\mathrm{E}$$

(85) the belief

$$\dfrac{\dfrac{}{} \quad \dfrac{\dfrac{}{\mathrm{N/?SP}} \quad \dfrac{}{\mathrm{?SP}}\,?\mathrm{I}}{\mathrm{N}}\,/\mathrm{E}}{\mathrm{NP}}\,/\mathrm{E}$$

5.2 Iterability

The structural rules of contraction and expansion are given in (86) (note that expansion is subsumed by weakening).

(86) a. $\Gamma\ Y \Rightarrow X \quad [\mathrm{C}]$
$\quad\Gamma\ Y\ Y \Rightarrow X$

b. $\Gamma\ Y\ Y \Rightarrow X \quad [\mathrm{E}]$
$\quad\Gamma\ Y \Rightarrow X$

The rules describe invariance of validities under numbers of occurrences of premisses. Such flexibility has linguistic analogy in e.g. the variable number of conjuncts that can occur left of a coordinator, and the variable number of gaps that can be filled by a fronted element in a language permitting parasitic extraction. Two possibilities are as follows: there may be a unary type constructor $^+$ indicating *some* degree of iteration, or one ! indicating *any* degree of iteration. The sequence $\Gamma(X^+)$ abbreviates the infinite disjunction $\Gamma(X)$ or $\Gamma(X\ X)$, or $\Gamma(X\ X\ X)$, etc. The sequence $\Gamma(!X)$ abbreviates the infinite conjunction $\Gamma(X)$ and $\Gamma(X\ X)$, and $\Gamma(X\ X\ X)$, etc. The following rules of deduction suggest themselves[4].

[4] Interactions of iteration and optionality are not considered here.

(87) a. $\dfrac{\begin{matrix}\vdots\\ X\end{matrix}}{X^+}{+}\mathrm{I}$ b. $\dfrac{\begin{matrix}\vdots\\ X^+\end{matrix} \quad \begin{matrix}\vdots\\ X^+\end{matrix}}{X^+}!\mathrm{E}$

(88) a. $\dfrac{\begin{matrix}\vdots\\ !X\end{matrix}}{X}$ b. $\dfrac{\begin{matrix}\vdots\\ !X\end{matrix}}{!X \quad !X}$

Correspondingly there are the following sequent rules:

(89) $\dfrac{\Gamma \Rightarrow X^+ \quad [^+\mathrm{R}]}{\Gamma \Rightarrow X}$ $\qquad$ $\dfrac{\Gamma(X^+\ X^+) \Rightarrow Y \quad [^+\mathrm{C}]}{\Gamma(X^+) \Rightarrow Y}$

(90) $\dfrac{\Gamma(!X) \Rightarrow Y \quad [!\mathrm{L}]}{\Gamma(X) \Rightarrow Y}$ $\qquad$ $\dfrac{\Gamma(!X) \Rightarrow Y \quad [!\mathrm{E}]}{\Gamma(!X\ !X) \Rightarrow Y}$

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

(91)
$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{John}}{\mathrm{NP}}}{\mathrm{NP}^+} \quad \dfrac{\dfrac{\text{Bill}}{\mathrm{NP}}}{\mathrm{NP}^+}}{\mathrm{NP}^+} \quad \dfrac{\dfrac{\text{Mary}}{\mathrm{NP}}}{\mathrm{NP}^+}}{\mathrm{NP}^+} \quad \dfrac{\dfrac{\text{and}}{(\mathrm{NP}^+\backslash\mathrm{NP})/\mathrm{NP}} \quad \dfrac{\text{Suzy}}{\mathrm{NP}}}{\mathrm{NP}^+\backslash\mathrm{NP}}/\mathrm{E}}{\mathrm{NP}}\backslash\mathrm{E}$$

5.3 Word Order Variation

The structural rule of exchange (or: permutation) is as follows:

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

Just as linear logic introduces weaken and contract structural modalities to replace the structural rules, we may propose to supply exchange or permute connectives to sequence logic. Let $\Gamma \triangleright X\ Y_1\ \ldots\ Y_n$ abbreviate the conjunction $\Gamma\ X\ Y_1\ \ldots\ Y_n$ and $\Gamma\ Y_1\ X\ \ldots\ Y_n$ ...and $\Gamma\ Y_1\ \ldots\ X\ Y_n$ and $\Gamma\ Y_1\ \ldots\ Y_n\ X$. Likewise, let $Y_1\ \ldots\ Y_n\ X \triangleleft\ \Gamma$ abbreviate $Y_1\ \ldots\ Y_n\ X\ \Gamma$ and $Y_1\ \ldots\ X\ Y_n\ \Gamma$ ...and $Y_1\ X\ \ldots\ Y_n\ \Gamma$ and $X\ Y_1\ \ldots\ Y_n\ \Gamma$. Then the following rules are valid:

(93) a. $\dfrac{\begin{matrix}\vdots\\ X\end{matrix}}{\triangleright \mathrm{X}}\triangleright\mathrm{I}$ b. $\dfrac{\begin{matrix}\vdots\\ X\end{matrix}}{\mathrm{X}\triangleleft}\triangleleft\mathrm{I}$

(94) a. $\dfrac{\begin{matrix}\vdots\\ \triangleright X\end{matrix} \quad \begin{matrix}\vdots\\ Y\end{matrix}}{Y \quad \triangleright X}$ b. $\dfrac{\begin{matrix}\vdots\\ Y\end{matrix} \quad \begin{matrix}\vdots\\ X\triangleleft\end{matrix}}{X \triangleleft \quad Y}$

(95) a. $\dfrac{\begin{matrix}\vdots\\ \rhd X\end{matrix}}{X}\rhd\mathrm{E}$ b. $\dfrac{\begin{matrix}\vdots\\ X\lhd\end{matrix}}{X}\lhd\mathrm{E}$

The rules of ▷I and ◁I are subject to the condition that every path from root to undischarged assumption contain a licensing modal type (right permute and left permute types respectively which do not depend on discharged assumptions). There are the following sequent rules, where ▷Γ and ◁Γ indicate sequences of the appropriate modal types. Note that the Left and Right rules are those for S4.

(96)

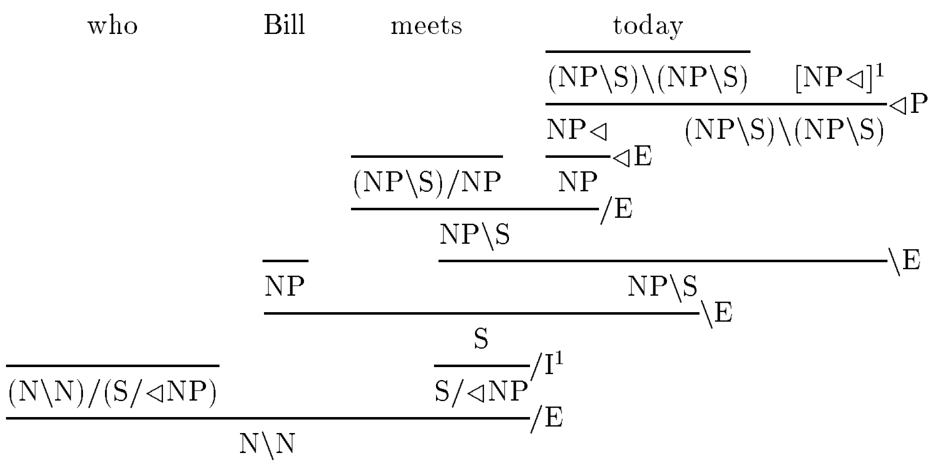
$$\frac{\rhd\Gamma \Rightarrow \rhd X}{\rhd\Gamma \Rightarrow X}[\rhd\mathrm{R}] \qquad \frac{\Gamma\lhd \Rightarrow X\lhd}{\Gamma\lhd \Rightarrow X}[\lhd\mathrm{R}]$$

$$\frac{\Gamma(\rhd X\ Y) \Rightarrow Z}{\Gamma(Y\ \rhd X) \Rightarrow Z}[\rhd\mathrm{P}] \qquad \frac{\Gamma(Y\ X\lhd) \Rightarrow Z}{\Gamma(X\lhd\ Y) \Rightarrow Z}[\lhd\mathrm{P}]$$

$$\frac{\Gamma(\rhd X) \Rightarrow Y}{\Gamma(X) \Rightarrow Y}[\rhd\mathrm{L}] \qquad \frac{\Gamma(X\lhd) \Rightarrow Y}{\Gamma(X) \Rightarrow Y}[\lhd\mathrm{L}]$$

Assignment of type (N\N)/(S/NP◁) to a relative pronoun will now allow it to fill a gap in *any* position, not just a clause-peripheral one:

(97)



In this way it is possible to *define* the kind of discontinuous type-constructors introduced in Moortgat (1989). Furthermore, assignment of a type (N\N)/(S/!(NP◁)) will allow it to fill *any number of* gaps in any positions. Thus "the paper which John files without reading" can be derived as follows:

(98) Suzy files without reading

$$
\begin{prooftree}
\AxiomC{}
\UnaryInfC{NP}
\AxiomC{}
\UnaryInfC{$(NP\backslash S)/NP$}
\AxiomC{}
\UnaryInfC{$((NP\backslash S)\backslash(NP\backslash S))/VP$}
\AxiomC{$!(NP\lhd)$}
\RightLabel{$!E$}
\UnaryInfC{$NP\lhd$}
\RightLabel{$\lhd P$}
\UnaryInfC{$NP\lhd \quad VP/NP$}
\RightLabel{$\lhd P$}
\BinaryInfC{$NP\lhd \quad ((NP\backslash S)\backslash(NP\backslash S))/VP$}
\RightLabel{$\lhd E$}
\UnaryInfC{NP}
\RightLabel{$/E$}
\BinaryInfC{$NP\backslash S$}
\AxiomC{$[!(NP\lhd)]^1$}
\noLine
\UnaryInfC{$!(NP\lhd)$}
\RightLabel{$!E$}
\UnaryInfC{$NP\lhd$}
\RightLabel{$\lhd E$}
\UnaryInfC{NP}
\RightLabel{$/E$}
\UnaryInfC{VP}
\RightLabel{$/E$}
\UnaryInfC{$(NP\backslash S)\backslash(NP\backslash S)$}
\RightLabel{$\backslash E$}
\BinaryInfC{$NP\backslash S$}
\RightLabel{$\backslash E$}
\BinaryInfC{S}
\RightLabel{$/I^1$}
\UnaryInfC{$S/!(NP\lhd)$}
\end{prooftree}
$$

The permute modalities that have been given signify that a type belongs *anywhere* to one side. Dual permute modalities can signify that a type belongs *somewhere* to one side. The logic may be as follows:

(99) a.
$$\frac{\begin{matrix}\vdots\\ X\end{matrix}}{X>}{>}I$$
b.
$$\frac{\begin{matrix}\vdots\\ X\end{matrix}}{<X}{<}I$$

(100) a.
$$\frac{\begin{matrix}\vdots & \vdots\\ X> & Y\end{matrix}}{Y \quad X>}$$
b.
$$\frac{\begin{matrix}\vdots & \vdots\\ Y & <X\end{matrix}}{<X \quad Y}$$

(101)
$$\frac{\Gamma \Rightarrow X>}{\Gamma \Rightarrow X}\;[>R] \qquad \frac{\Gamma \Rightarrow <X}{\Gamma \Rightarrow X}\;[<R]$$

$$\frac{\Gamma(X> \;\; Y) \Rightarrow Z}{\Gamma(Y \;\; X>) \Rightarrow Z}\;[>P] \qquad \frac{\Gamma(Y \;\; <X) \Rightarrow Z}{\Gamma(<X \;\; Y) \Rightarrow Z}\;[<P]$$

$$\frac{Y_1 \ldots Y_n X > \Gamma \Rightarrow Z}{\begin{matrix} Y_1 \ldots Y_n X \Gamma \Rightarrow Z\\ Y_1 \ldots X Y_n \Gamma \Rightarrow Z\\ \vdots\\ Y_1 X \ldots Y_n \Gamma \Rightarrow Z\\ X Y_1 \ldots Y_n \Gamma \Rightarrow Z\end{matrix}}\;[>L] \qquad \frac{\Gamma < X Y_1 \ldots Y_n \Rightarrow Z}{\begin{matrix} \Gamma X Y_1 \ldots Y_n \Rightarrow Z\\ \Gamma Y_1 X \ldots Y_n \Rightarrow Z\\ \vdots\\ \Gamma Y_1 \ldots X Y_n \Rightarrow Z\\ \Gamma Y_1 \ldots Y_n X \Rightarrow Z\end{matrix}}\;[<L]$$

Then for example a type $X/<Y$ represents elements which are functors over Xs *somewhere* to the right, but not necessarily immediately to the right. 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 device illustrated in section 4, but the essential idea is as illustrated in (102).

(102)

$$\dfrac{\dfrac{\dfrac{\text{ring}}{\text{(VP/Prt)/<NP}} \qquad \dfrac{\dfrac{\text{up}}{\text{Prt}} \quad \dfrac{\dfrac{\text{John}}{\text{NP}}}{\text{<NP}}{<}I}{\text{<NP} \quad \text{Prt}}{<}P}{\text{VP/Prt}}/E \qquad \text{Prt}}{\text{VP}}/E$$

6 Conclusion

The paper has presented an extension of categorial grammar to a more general type theoretic formalism akin to linear logic. In addition to further development of the kind illustrated here, several tasks arise: precise formulation of the logic and determination of its properties, specification of a formal semantics, and the development of effective automated type inference. No doubt much of relevance is to be found in the Computer Science literature on type theory and linear logic. This paper at least aims to have provided a survey of the possibilities for practical description of natural language in the categorial tradition.

References

Ades, Anthony E. and Mark J. Steedman: 1982, 'On the Order of Words', *Linguistics and Philosophy* **4**, 517–558.

Ajdukiewicz, Kazimierz: 1935, 'Die syntaktische Konnexitat', *Studia Philosophica* **1**, 1–27, translated in S. McCall (ed.), *Polish Logic: 1920–1939*, Oxford University Press, Oxford, pp. 207–231.

Bar-Hillel, Yehoshua: 1953, 'A quasi-arithmetical notation for syntactic description', *Language* **29**, 47–58.

van Benthem, Johan: 1983, 'The semantics of Variety in Categorial Grammar', Report 83-29, Department of Mathematics, Simon Fraser University. Also in W. Buszkowski *et al.* (eds.), 1988, *Categorial Grammar*, Volume 25, Linguistic & Literary Studies in Eastern Europe, John Benjamins, Amsterdam/Philadelphia, pp. 37–55.

van Benthem, Johan:1987, 'Categorial Grammar and Type Theory', Prepublication Series 87–07, Institute for Language, Logic and Information, University of Amsterdam.

van Benthem, Johan: 1989a, 'Language in Action', Prepublication Series LP–89–04, Institute for Language, Logic and Information, University of Amsterdam.

van Benthem, Johan: 1989b, 'Categorial Grammar Meets Unification', ms., Institute for Language, Logic and Information, University of Amsterdam.

Buszkowski, W.: 1988, 'Three theories of categorial grammar', in W. Buszkowski *et al.* (eds.), 1988, *Categorial Grammar*, Volume 25, Linguistic & Literary Studies in Eastern Europe, John Benjamins, Amsterdam/Philadelphia, pp. 57–84.

Desclés, Jean-Pierre, Zlatka Guentchéva and Sebastian Shaumyan: 1986, 'Theoretical Analysis of Reflexivisation in the Framework of Applicative Grammar', *Lingvisticæ Investigationes* **X:1**, 1–65.

Girard, Jean-Yves: 1989, 'Towards a Geometry of Interaction', *Proceedings of the AMS Conference on Categories, Logic and Computer Science.*

Girard, Jean-Yves, Paul Taylor, and Yves Lafont: 1989, *Proofs and Types*, Volume 7, Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, Cambridge.

Hobbs, Jerry R. and Stanley J. Rosenschein: 1978, 'Making Computational Sense of Montague's Intensional Logic', *Artifical Intelligence*, **9**, 287–306.

Keenan, Edward L. and Alan Timberlake: 1988, 'Natural Language Motivations for Extending Categorial Grammar' in R. Oehrle, E. Bach and D. Wheeler (eds.) *Categorial Grammars and Natural Language Structures*, D. Reidel, Dordrecht, pp. 265–295.

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

Lambek, J.: 1961, 'On the calculus of syntactic types', in *Structure of language and its mathematical aspects*, Proceedings of the Symposia in Applied Mathematics **XII**, American Mathematical Society.

Moortgat, Michael: 1989, *Categorial Investigations: Logical and Linguistic Aspects of the Lambek Calculus*, Foris, Dordrecht.

Morrill, Glyn: 1989a, 'Intensionality, Boundedness, and Modal Logic', Research Paper EUCCS/RP-32, Centre for Cognitive Science, University of Edinburgh.

Morrill, Glyn: 1989b, 'Grammar as Logic', Research Paper EUCCS/RP–34, Centre for Cognitive Science, University of Edinburgh.

Morrill, Glyn: to appear, 'Intensionality and Boundedness', *Linguistics and Philosophy.*

Morrill, Glyn, Neil Leslie, Mark Hepple, and Guy Barry (1990), 'Categorial Deductions and Structural Operations', ms. Centre for Cognitive Science, University of Edinburgh

Oehrle, Richard T.:1988, 'Multi-Dimensional Compositional Functions as a Basis for Grammatical Analysis', in R. Oehrle, E. Bach and D. Wheeler (eds.) *Categorial Grammars and Natural Language Structures*, D. Reidel, Dordrecht, pp. 349–389.

Pollard, Carl and Ivan A. Sag *Information-Based Syntax and Semantics, Volume 1*, Number 13, CSLI Lecture Notes, Center for the Study of Language and Information, Stanford.

Prawitz, D. 1965. *Natural Deduction*: *A Proof- Theoretical Study.* Almqvist and Wiksell, Uppsala.

Szabolcsi, Anna: 1987, 'Bound Variables in Syntax', Proceedings of the Sixth Amsterdam Colloquium, Institute for Language, Logic and Information, University of Amsterdam, pp.331–351.

Wood, Mary McGee: 1988, 'A Categorial Syntax for Coordinate Constructions', Ph.D. thesis, University of London. Technical Report UMCS–89–2–1, Department of Computer Science, University of Manchester.

Zeevat, Henk, Ewan Klein, and Jo Calder: 1987, 'Unification Categorial Grammar', in N.

Haddock, E. Klein, and G. Morrill (eds.) *Categorial Grammar, Unification Grammar, and Parsing*, Volume 1, Edinburgh Working Papers in Cognitive Science, Centre for Cognitive Science, University of Edinburgh.