Rules and Derivation: Binding Phenomena and Coordination in Categorial Logic

Glyn Morrill
Centre for Cognitive Science, University of Edinburgh
OTS, University of Utrecht
CWI, University of Amsterdam

DYANA
Deliverable R1.2.D
December 1990

DYANA
Dynamic Interpretation of Natural Language

ESPRIT Basic Research Action BR3175

University of Edinburgh
Centre for Cognitive Science
Department of Artificial Intelligence
Centre for Speech Technology Research
Department of Linguistics
Department of Computer Science

Universiteit van Amsterdam
Instituut voor Taal, Logica en Informatie

Universität Stuttgart
Institut für maschinelle Sprachverarbeitung

Universität Tübingen
Seminar für natürlich-sprachliche Systeme

Universität München
Zentrum für Informations und Sprachforschung

For copies of reports, updates on project activities and other DYANA-related information, contact

DYANA Administrator
Centre for Cognitive Science
University of Edinburgh
2, Buccleuch Place
Edinburgh EH8 9LW, UK

Contents

1 Introduction

The structure of this paper is as follows. In section 1 there is set up a general notion of *free generative grammar* (FGG); this is approximately the narrowest conception of grammar broad enough to encompass the varying cases of later sections. Significantly, it is noted that there may be different presentations of the same grammar, differing in their axioms and rules, but generating the same theory. It will be a theme throughout that decision procedures for theorem proving (= derivation search) are obtained by partially executing axioms and rules with respect to Cut to obtain presentations for which Cut is eliminable. The idea of FGG and Cut-free presentation for derivation search is exemplified in section 2 by context-free phrase structure grammar, and an extension thereof designed to characterise relativisation and quantification. It is shown how island constraints may be captured by a system generating partially bracketed sequences, cf. weak generative capacity (sequences) and strong generative capacity (trees). In section 3 there is discussed categorial grammar: the associative (sequences) and non-associative (trees) Lambek calculi, and more general categorial logics. A grammar is given capturing various constraints on and between relativisation, quantification, reflexivisation and coordination.

This research is carried out in a context in which essentially Montagovian analysis of phenomena such as quantification and coordination has been approached via systems of unary rules effecting flexibility in the types which expressions adopt. Such approaches raise computational issues: finding terminating algorithms for derivation search, and finding efficient such algorithms, e.g. by avoiding recomputation of equivalent derivations. The type-shifting approach to quantification is well-exemplified by Hendriks (1990), and Moortgat (1990) shows how decision procedures for a largely equivalent theory can be obtained for categorial formalisation employing an 'exponentiation' type-constructor, by Cut-free derivation search. The present paper integrates and extends categorial coverage, illustrating how systems of type-shift can be provided with a principled basis in terms of categorial logic, and hence be provided with terminating algorithms for derivation search. Consideration here however does not extend from decidablity to efficiency.

2 Free Generative Grammar

Let us assume that a mathematical model of a language will consist of various language objects standing in various relations. We say the following:

(1) A *language model structure* consists of a domain of *language objects* including a subset of *basic language objects*, and a set of operations over that domain.

A language object will generally be a tuple made up of values in algebras for whatever linguistic dimensions are under consideration; an operation will be a tuple of operations over the individual algebras (cf. Oehrle 1988). The instance implicit in later sections has it that language objects are form-meaning pairs (signs) classified into categories including a distinguished category of sentences, the meanings of which are the subject of a consequence relation. However, the framework of free generative grammar defined here abstracts over such details.

The category symbols (types) $A, B, C, \ldots$, will be interpreted as sets of objects in language model structures:

(2) An *inhabitation* is a relation between the domain of language objects and the set of types.

A lexical inhabitation is a relation between the basic language objects and the set of types.

(3) An (n-ary) *configuration* is a length n bracketed sequence of types.

Configurations $X, Y, Z, \ldots$ include those with no brackets, those with binary bracketing, and so on. We write $X(Y)$ for a configuration X with a distinguished subpart Y.

(4) A *statement of formation* consists of an n-ary operation θ on the domain of language objects, an n-ary configuration X, and a type A.

A statement of formation asserts that applying the operation θ to any language objects in the types X yields a language object in the type A. It may be true or false with respect to an inhabitation. In the following, the operations on forms will uniformly be left implicit as left-to-right concatenation of the configuration type forms; this is a convenient but not necessary feature.

(5) A *rule of formation* consists of a set $\{\Gamma_1, \ldots, \Gamma_n\}$ of premise statements of formation, and a conclusion statement of formation Δ.

We write: $\frac{\Gamma_1 \ldots \Gamma_n}{\Delta}$. A rule of formation is said to be axiomatic if and only if it has an empty premise set. There are two universally valid rules of formation, id and Cut, interpreted by the identity and composition operations:

(6) $A \Rightarrow A$ [id] $\dfrac{Y \Rightarrow B \quad X(B) \Rightarrow A}{X(Y) \Rightarrow A}$ [Cut]

(7) A *derivation* is a sequence of statements of formation each of which is either axiomatic, or else is the conclusion of a rule of formation the premises of which occur earlier in the sequence.

A set of rules of formation is said to generate the set of all the statements of formation which can conclude derivations on the basis of those rules.

(8) We call a *theory of formation* any set of statements of formation closed under id and Cut. The members of a theory of formation are called *theorems of formation.*

A presentation of a theory of formation is a set of rules of formation generating that theory.

(9) A *grammar* consists of a lexical inhabitation and a theory of formation.

A grammar presentation consists of a lexical inhabitation and a presentation of a theory of formation.

3 Phrase Structure Grammar

Consider the following presentation of a phrase structure grammar theory of formation annotated with a simple functional semantics (' represents application explicitly).

(10) $x : A \Rightarrow x : A$ [id] $\dfrac{Y \Rightarrow \beta{:}B \quad X(y{:}B) \Rightarrow \alpha{:}A}{X(Y) \Rightarrow \alpha[\beta/y]{:}A}$ [Cut]

$x{:}\mathrm{NP}\ y{:}\mathrm{VP} \Rightarrow y\,'\,x{:}\mathrm{S}$ [r1]
$x{:}\mathrm{D}\ y{:}\mathrm{CN} \Rightarrow x\,'\,y{:}\mathrm{NP}$ [r2]
$x{:}\mathrm{TV}\ y{:}\mathrm{NP} \Rightarrow x\,'\,y{:}\mathrm{VP}$ [r3]
$x{:}\mathrm{SV}\ y{:}\mathrm{S} \Rightarrow x\,'\,y{:}\mathrm{VP}$ [r4]

A *derivation* will be a list of statements of formation each of which is either an axiom, or else follows from carlier statements of formation by rule. In relation to (11) there is (12); underlines indicate Cut types.

(11) *John thinks the man loves Mary*

(12)	1.	$\mathrm{D\ CN} \Rightarrow \underline{\mathrm{NP}}$	r2
	2.	$\underline{\mathrm{NP}}\ \mathrm{VP} \Rightarrow \mathrm{S}$	r1
	3.	$\mathrm{D\ CN}\ \underline{\mathrm{VP}} \Rightarrow \mathrm{S}$	1, 2, Cut
	4.	$\mathrm{NP}\ \underline{\mathrm{VP}} \Rightarrow \mathrm{S}$	r1
	5.	$\mathrm{SV\ S} \Rightarrow \underline{\mathrm{VP}}$	r4
	6.	$\mathrm{TV\ NP} \Rightarrow \underline{\mathrm{VP}}$	r3
	7.	$\mathrm{D\ CN\ TV\ NP} \Rightarrow \underline{\mathrm{S}}$	6, 3, Cut
	8.	$\mathrm{NP\ SV}\ \underline{\mathrm{S}} \Rightarrow \mathrm{S}$	5, 4, Cut
	9.	$\mathrm{NP\ SV\ D\ CN\ TV\ NP} \Rightarrow \mathrm{S}$	7, 8, Cut

Derivations are summarised by derivation structures which identify equivalent derivations, i.e. ones mapping from the same configuration to the same type by the same operation.

(13)

$$\dfrac{\dfrac{\text{John}}{x{:}\mathrm{NP}} \qquad \dfrac{\dfrac{\text{thinks}}{y{:}\mathrm{SV}} \qquad \dfrac{\dfrac{\dfrac{\text{the}}{z{:}\mathrm{D}} \quad \dfrac{\text{man}}{w{:}\mathrm{CN}}}{z\,'\,w{:}\mathrm{NP}}\,\mathrm{r2} \qquad \dfrac{\dfrac{\text{loves}}{u{:}\mathrm{TV}} \quad \dfrac{\text{Mary}}{v{:}\mathrm{NP}}}{u\,'\,v{:}\mathrm{VP}}\,\mathrm{r3}}{(u\,'\,v)\,'\,(z\,'\,w){:}\mathrm{S}}\,\mathrm{r1}}{y\,'\,((u\,'\,v)\,'\,(z\,'\,w)){:}\mathrm{VP}}\,\mathrm{r4}}{(y\,'\,((u\,'\,v)\,'\,(z\,'\,w)))\,'\,x{:}\mathrm{S}}\,\mathrm{r1}$$

The annotated statements of formation show how to construct the meanings of compound expressions in terms of the meanings of their parts. We obtain a schematic meaning representation into which we can substitute the meaning representations for words given in the lexicon:

(14)	*John*	– john	:= NP
	Mary	– mary	:= NP
	walks	– walk	:= VP
	man	– man	:= CN
	the	– the	:= D
	thinks	– think	:= SV
	likes	– like	:= TV

Thus (13) associates the following meaning with (11).

(15) (think ’((the ’ man) ’(love ’ mary))) ’john

By *derivation search* with respect to a theory of formation we mean the task of finding all the operations mapping from a given configuration to a given type in that theory. This is tantamount to parsing. A *presentation* of a theory of formation offers a strategy for derivation search: simply search backwards from the given configuration and type for all derivations which have that statement of formation as conclusion.

Presentations including Cut however provide a poor basis for such derivation search: the type which is Cut out is introduced as an unknown on every backwards application of Cut. A better strategy is to perform derivation search with a presentation which is Cut-free. The Cut rule must be admissable in such presentations since it is universal and all FGG theories of formation are closed under Cut. But the idea is to find presentations in which the contextual management of Cut is built into the other rules, e.g. we can try partially executing the axiomatic rules above with respect to Cut. Two ways of doing this lead to the classical bottom-up and top-down procedures, of which the later is as follows:

(16) $x : A \Rightarrow x : A$ [id]

$X\ Y \Rightarrow \beta\ '\ \alpha{:}\mathrm{S}$ [R1]
$X \Rightarrow \alpha{:}\mathrm{NP}$
$Y \Rightarrow \beta{:}\mathrm{VP}$

$X\ Y \Rightarrow \alpha\ '\ \beta{:}\mathrm{NP}$ [R2]
$X \Rightarrow \alpha{:}\mathrm{D}$
$Y \Rightarrow \beta{:}\mathrm{CN}$

$X\ Y \Rightarrow \alpha\ '\ \beta{:}\mathrm{VP}$ [R3]
$X \Rightarrow \alpha{:}\mathrm{TV}$
$Y \Rightarrow \beta{:}\mathrm{NP}$

$X\ Y \Rightarrow \alpha\ '\ \beta{:}\mathrm{VP}$ [R4]
$X \Rightarrow \alpha{:}\mathrm{SV}$
$Y \Rightarrow \beta{:}\mathrm{S}$

The theorem of formation considered earlier is now derived as follows.

(17) NP SV D CN TV NP ⇒ S [R1]
NP ⇒ NP [id]
SV D CN TV NP ⇒ VP [R4]
SV ⇒ SV [id]
D CN TV NP ⇒ S [R1]
D CN ⇒ NP [R2]
D ⇒ D [id]
CN ⇒ CN [id]
TV NP ⇒ VP [R3]
TV ⇒ TV [id]
NP ⇒ NP [id]

The presentation forms a good basis for derivation search since goal-driven search induces no new unknowns in subgoals.

From this review of technical and computational properties of phrase structure grammar from the FGG point of view, we turn in the next section to empirical matters and their treatment by augmentation of PSG with non-axiomatic rules of formation, intermediaries betwen metarules (cf. Generalised Phrase Structure Grammar) and transformations (cf. Transformational Grammar and Government-Binding).

3.1 Relativisation and Quantification

Consider the following.

(18) a. the man that walks
b. *the man that walk
c. *the men that walks
d. the men that walk

(19) the man that Suzy likes

(20) a. the man that Mary thinks walks
b. *the man that Mary thinks walk
c. *the men that Mary thinks walks
d. the men that Mary thinks walk

(21) the man that Mary thinks Bill claims Suzy likes

For relativisation, let us make the assumption that elements of types forming a sentence are suitable to form the body of a *that*-relative clause when a noun phrase type is omitted, and let the meaning of this incomplete sentence be given by the lambda abstraction over the meaning of the missing element. This is effected by the following rule which is schematised over sequences of types.[1]

(22) $\dfrac{X\ x{:}\mathrm{NP}\ Y \Rightarrow \gamma{:}\mathrm{S}}{X\ Y \Rightarrow \lambda x\gamma{:}\mathrm{RB}}\mathrm{R}$

(23) $x{:}\mathrm{CN}\ y{:}\mathrm{RC} \Rightarrow y\ '\ x{:}\mathrm{CN}$ [r5]

(24) $x{:}\mathrm{RP}\ y{:}\mathrm{RB} \Rightarrow x\ '\ y{:}\mathrm{RC}$ [r6]

(25) *that* $-\ \lambda x\lambda y\lambda z((x\ '\ z) \wedge (y\ '\ z))$:= RP

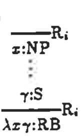
The rule is something in between a metarule and a transformation. Metarules (see e.g. Gazdar 1981, Gazdar et al. 1985, Morrill 1988) technically only close the axioms, with Cut

[1] Uniformly in what follows, parentheses are omitted under a convention whereby unary operators, including quantifier and lambda bindings, adhere more tightly than binary operators.

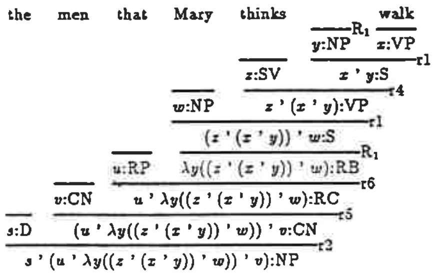
being left implicit as the only non-axiomatic rule of formation; metarules do not apply to theorems generated by Cut.[2] Transformations on the other hand apply to the tree structures generated by phrase structure grammar; these are distinct from statements of formation. We now actually turn to consider how derivation structures can record non-axiomatic inferences.

A derivation involving R is summarised by a derivation structure which is 'grown' at both root and leaf:

(26)



(27)



The result of substituting the lexical semantics into the derivational semantics and simplifying is:

(28) **the** ' λz(((**think** ' (**walk** ' z)) ' **mary**) ∧ (**men** ' z))

Montague's analysis of quantification allows quantifier phrases to take scope at the level of any dominating sentence; thus (29) is predicted to have two readings, and (30) three.

(29) *Every man loves a woman*
- ∀y((**man** ' y) → ∃x((**woman** ' x) ∧ ((**love** ' x) ' y)))
- ∃x((**woman** ' x) ∧ ∀y((**man** ' y) → ((**love** ' x) ' y)))

(30) *Every man claims a woman walks*
- ∀x((**man** ' x) → ((**claim** ' ∃y((**woman** ' y) ∧ (**walk** ' y))) ' x))
- ∀x((**man** ' x) → ∃y((**woman** ' y) ∧ ((**claim** ' (**walk** ' y)) ' x))
- ∃y((**woman** ' y) ∧ ∀x((**man** ' x) → ((**claim** ' (**walk** ' y)) ' x)))

For quantification, assume the aim is to allow quantifier phrases to occur at the same sites as other noun phrases, and to have them take scope at the level of any superordinate sentence:

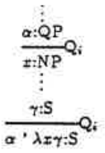
(31) $\dfrac{X \;\; x{:}\mathrm{NP} \;\; Y \Rightarrow \beta{:}\mathrm{S}}{X \;\; \gamma{:}\mathrm{QP} \;\; Y \Rightarrow \gamma \text{ ' } \lambda x\beta{:}\mathrm{S}}\mathrm{Q}$

(32) x:Q y:CN ⇒ x ' y:QP [r7]

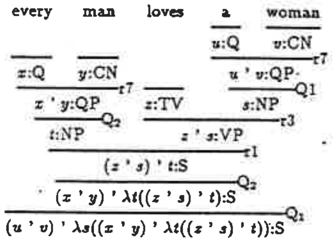
(33) *every* – λxλy∀z((x ' z) → (y ' z)) := Q
a – λxλy∃z((x ' z) ∧ (y ' z)) := Q

Derivations involving quantification are summarised by derivation structures extending at the root a derivation of S from NP, and connecting a derivation of QP at the NP leaf:

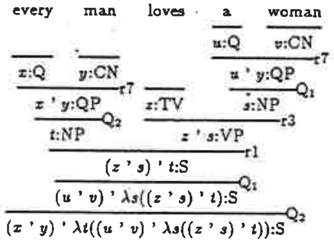
(34)



(35)



(36)



Substitution of lexical semantics and simplification for (35) gives (37a) and that for (36) gives (37b).

(37) a. ∃x((**woman** ' x) ∧ ∀y((**man** ' y) → ((**love** ' x) ' y)))
b. ∀y((**man** ' y) → ∃x((**woman** ' x) ∧ ((**love** ' x) ' y)))

In a similar way, three readings are obtained for *every man claims a woman sings*: one with the existential taking scope at the level of the embedded sentence, and two others with alternative quantifier scoping at the matrix level.

[2] For example, if A B ⇒ C and D ⇒ B are axiomatic rules of formation then A D ⇒ C is a theorem of formation, but generated by Cut, this may not be input to a metarule.

The top-down Cut-free presentation of the current grammar suitable for derivation search is given by adding the following to (16).

(38) $X\ Y\ \Rightarrow \lambda x\gamma{:}\mathrm{RB}$ [REL]
$\quad X\ x{:}\mathrm{NP}\ Y\ \Rightarrow \gamma{:}\mathrm{S}$

$X\ Y\ \Rightarrow \beta\ '\ \alpha{:}\mathrm{CN}$ [R5]
$\quad X\ \Rightarrow \alpha{:}\mathrm{CN}$
$\quad Y\ \Rightarrow \beta{:}\mathrm{RC}$

$X\ Y\ \Rightarrow \alpha\ '\ \beta{:}\mathrm{RC}$ [R6]
$\quad X\ \Rightarrow \alpha{:}\mathrm{RP}$
$\quad Y\ \Rightarrow \beta{:}\mathrm{RB}$

$X\ Y\ Z\ \Rightarrow \alpha\ '\ \lambda x\gamma{:}\mathrm{S}$ [QUANT]
$\quad Y\ \Rightarrow \alpha{:}\mathrm{QP}$
$\quad X\ x{:}\mathrm{NP}\ Z\ \Rightarrow \gamma{:}\mathrm{S}$

$X\ Y\ \Rightarrow \alpha\ '\ \beta{:}\mathrm{QP}$ [R7]
$\quad X\ \Rightarrow \alpha{:}\mathrm{Q}$
$\quad Y\ \Rightarrow \beta{:}\mathrm{CN}$

3.2 Brackets for Islands

The present system however fails to capture the islandhood of relative clauses: the extraction from a relative clause in (39) is ungrammatical (cf. Complex Noun Phrase Constraint, CNPC).

(39)*the man that Mary likes the woman that loves

The example has a top-down Cut-free derivation involving the derivation of *Mary likes the woman that loves* as a relative body as follows:[3]

[3] In fact there are two derivations, with the relative pronouns alternatively binding the subject and object of *loves*, but the subsequent remarks apply equally to both cases.

```
(40)  NP TV D CN RP TV ⇒ RB                              [REL]
         NP TV D CN RP TV NP ⇒ S                         [R1]
            NP ⇒ NP                                      [id]
            TV D CN RP TV NP ⇒ VP                        [R3]
               TV ⇒ TV                                   [id]
               D CN RP TV NP ⇒ NP                        [R2]
                  D ⇒ D                                  [id]
                  CN RP TV NP ⇒ CN                       [R5]
                     CN ⇒ CN                             [id]
                     RP TV NP ⇒ RC                       [R6]
                        RP ⇒ RP                          [id]
                        TV NP ⇒ RB                       [REL]
                           NP TV NP ⇒ S                  [R1]
                              NP ⇒ NP                    [R1]
                              TV NP ⇒ VP                 [R3]
                                 TV ⇒ TV                 [id]
                                 NP ⇒ NP                 [id]
```

Furthermore, while the grammar predicts that (41) has two readings, in fact quantifiers cannot take scope outside of relative clauses in which they occur.[4]

(41) A man that likes every woman sings.

The adjustment invoked here involves a move from the flat configurations used so far, to bracketed configurations. Briefly, relative pronouns will require their clauses to be bracketed off as islands: floors to relativisation and ceilings to quantification. The theorems of formation will now include partially bracketed statements of formation obtained by replacing r6 and R6 as follows:

(42) $[x{:}\mathrm{RP}\ y{:}\mathrm{RB}]\ \Rightarrow x\ '\ y{:}\mathrm{RC}$ [r6′]

(43) $[X\ Y]\ \Rightarrow \alpha\ '\ \beta{:}\mathrm{RC}$ [R6′]
$\quad X\ \Rightarrow \alpha{:}\mathrm{RP}$
$\quad Y\ \Rightarrow \beta{:}\mathrm{RB}$

Now, for R6′ to go in place of R6 in (40), the target statement of formation would need a bracketed configuration: NP TV D CN [RP TV]. But then as the rule has been given in (22) — with NP not permitted to be embedded within brackets — the top rule application REL could not place the 'trace' NP *inside* these brackets, where the trace must be in order to combine with the embedded TV. These conflicting requirements block derivation. A similar line of argument shows why as the quantifier rule (31) has been given, this move also blocks quantifier 'raising' out of a relative clause.

[4] Fodor and Sag 1982 observe that indefinites do in fact always have readings in which the quantifier is construed outermost at the matrix level, violating any island constraints, but they argue that such readings arise from a quantificational/referential ambiguity of indefinites.

4 Categorial Grammar

The previous section illustrated FGG, derivation search by means of a Cut-free presentation of the theory of formation, and use of bracketed configurations to capture islandhood. The PSG framework used there is minimally lexical in that a particular language model is defined by setting up, in addition to the lexicon, a theory of formation subject only to the laws of id and Cut which are everywhere valid in FGG. We will now switch to the CG framework which is maximally lexical in that a particular language model is defined soley by setting up a lexicon. The theory of formation is universal, being just the logic expressing the intended semantics of the type-constructors used.

4.1 The Lambek Calculi

Classical categorial grammar AB has the following Cut-based presentation:

(44) $x : A \Rightarrow x : A$ [id] $\dfrac{Y \Rightarrow \beta{:}B \quad X(y{:}B) \Rightarrow \alpha{:}A}{X(Y) \Rightarrow \alpha[\beta/y]{:}A}$ [Cut]

$x : A/B \; y : B \Rightarrow x\,'\,y : A$ [/E] $y : B \; x : B\backslash A \Rightarrow x\,'\,y : A$ [\E]

A lexicon parallel in effect to the initial phrase structure grammar is as follows:

(45)
John – john := NP
Mary – mary := NP
walks – walk := NP\S
man – man := CN
the – the := NP/CN
thinks – think := (NP\S)/S
likes – like := (NP\S)/NP

AB derivations are represented by natural deduction style derivation structures. First, a lone type is a derivation (corresponding to id), then derivation structures are built thus:

(46)
$$\dfrac{\overset{\vdots}{A/B} \quad \overset{\vdots}{B}}{A}/\mathrm{E} \qquad\qquad \dfrac{\overset{\vdots}{B} \quad \overset{\vdots}{B\backslash A}}{A}\backslash\mathrm{E}$$

(47)
$$\dfrac{\dfrac{\text{John}}{\mathrm{NP}} \quad \dfrac{\dfrac{\text{thinks}}{(\mathrm{NP\backslash S})/\mathrm{S}} \quad \dfrac{\dfrac{\dfrac{\text{the}}{\mathrm{NP/CN}} \quad \dfrac{\text{man}}{\mathrm{CN}}}{\mathrm{NP}}/\mathrm{E} \quad \dfrac{\dfrac{\text{loves}}{(\mathrm{NP\backslash S})/\mathrm{NP}} \quad \dfrac{\text{mary}}{\mathrm{NP}}}{\mathrm{NP\backslash S}}/\mathrm{E}}{\mathrm{S}}\backslash\mathrm{E}}{\mathrm{NP\backslash S}}/\mathrm{E}}{\mathrm{S}}\backslash\mathrm{E}$$

Associative categorial grammar (ACG; Lambek 1958) is obtained by adding to this rules of conditionalisation:

(48) $\dfrac{X \; y{:}B \Rightarrow \alpha{:}A}{X \Rightarrow \lambda y\alpha{:}A/B}$ [/I] $\dfrac{y{:}B \; X \Rightarrow \alpha{:}A}{X \Rightarrow \lambda y\alpha{:}B\backslash A}$ [\I]

The corresponding derivation structures indicate cancellation of conditionalised peripheral types B in the subderivations of A.

(49)
$$\dfrac{\overset{\vdots}{A} \quad \overline{B}^{n}}{A/B}/\mathrm{I}_n \qquad\qquad \dfrac{\overline{B}^{n} \quad \overset{\vdots}{A}}{B\backslash A}\backslash\mathrm{I}_n$$

ACG theorems are unbracketed sequences of types. Non-associative categorial grammar (NCG; Lambek 1961) generates binary bracketed sequences of types:

(50) $x : A \Rightarrow x : A$ [id] $\dfrac{Y \Rightarrow \beta{:}B \quad X(y{:}B) \Rightarrow \alpha{:}A}{X(Y) \Rightarrow \alpha[\beta/y]{:}A}$ [Cut]

$[x : A \oslash B \; y : B] \Rightarrow x\,'\,y : A$ [$\oslash$E] $[y : B \; x : B \obslash A] \Rightarrow x\,'\,y : A$ [$\obslash$E]

$\dfrac{[X \; y{:}B] \Rightarrow \alpha{:}A}{X \Rightarrow \lambda y\alpha{:}A\oslash B}$ [$\oslash$I] $\dfrac{[y{:}B \; X] \Rightarrow \alpha{:}A}{X \Rightarrow \lambda y\alpha{:}B\obslash A}$ [$\obslash$I]

In alternative presentations (cf. Lambek, Zielonka) the rules of conditionalisation which schematise over unboundedly long sequences of types get compacted into unary rules of type shift.

(51) $A \Rightarrow A$ [id] $\dfrac{Y \Rightarrow B \quad X(B) \Rightarrow A}{X(Y) \Rightarrow A}$ [Cut]

$A/B \; B \Rightarrow A$ [/E] $B \; B\backslash A \Rightarrow A$ [\E]

$B \Rightarrow A/(B\backslash A)$ [L] $B \Rightarrow (A/B)\backslash A$ [L]

$A/B \Rightarrow (A/C)/(B/C)$ [D] $B\backslash A \Rightarrow (C\backslash B)\backslash(C\backslash A)$ [D]

$\dfrac{A \Rightarrow B \quad C \Rightarrow D}{A/D \Rightarrow B/C}$ [/] $\dfrac{A \Rightarrow B \quad C \Rightarrow D}{D\backslash A \Rightarrow C\backslash B}$ [\]

(52) $A \Rightarrow A$ [id] $\dfrac{Y \Rightarrow B \quad X(B) \Rightarrow A}{X(Y) \Rightarrow A}$ [Cut]

$[A \oslash B \; B] \Rightarrow A$ [$\oslash$E] $[B \; B \obslash A] \Rightarrow A$ [$\obslash$E]

$B \Rightarrow A \oslash (B \obslash A)$ [L] $B \Rightarrow (A \oslash B) \obslash A$ [L]

$\dfrac{A \Rightarrow B \quad C \Rightarrow D}{A \oslash D \Rightarrow B \oslash C}$ [$\oslash$] $\dfrac{A \Rightarrow B \quad C \Rightarrow D}{D \obslash A \Rightarrow C \obslash B}$ [$\obslash$]

The Cut-based unary rule presentations provide a poor basis for derivation search. Notice for example that simple application can be derived an infinite number of ways by complementary repeated lifting of functor and argument. What is desired is a notion of normal derivation such that backward-search for normal derivations (cf. Hepple and Morrill 1989) of a given statement of formation is complexity-reducing at every step. This is precisely what is provided by Buszkowski (1986) for the non-associative case. Note first that (ignoring id) every unary theorem of formation is either complexity increasing or complexity decreasing, where the metric of complexity is the number of type-constructor occurrences. Buszkowski shows that every Cut-based unary NCG derivation has a normal form counterpart derivation, which is obtained by applying top-down-complexity-reducing unary theorems top-down, and bottom-up-complexity-reducing unary theorems bottom-up, with a central core of binary applications. This normalisation result is used to show equal weak generative power of the non-associative calculus with context-free grammar. A corresponding normalisation for the associative calculus has not yet been demonstrated; it seems possible that if it were, it would bring with it the conjectured context-free weak equivalence result. But in its absence, the unary rule formulation does not provide the basis for a terminating algorithm for derivation search.

Rather, such an algorithm is obtained on the basis of a Cut-free presentation; this is how Lambek demonstrated the decidability of his calculi.

(53) $x : A \Rightarrow x : A$ [id]

$$\begin{array}{llll} X \Rightarrow \lambda y\alpha : A/B & [/R] & X(z : A/B\ Y) \Rightarrow \gamma[z\ \text{'}\ \beta/x] : C & [/L] \\ \quad X\ y : B \Rightarrow \alpha : A & & \quad Y \Rightarrow \beta : B & \\ & & \quad X(x : A) \Rightarrow \gamma : C & \end{array}$$

$$\begin{array}{llll} X \Rightarrow \lambda y\alpha : B\backslash A & [\backslash R] & X(Y\ z : B\backslash A) \Rightarrow \gamma[z\ \text{'}\ \beta/x] : C & [\backslash L] \\ \quad y : B\ X \Rightarrow \alpha : A & & \quad Y \Rightarrow \beta : B & \\ & & \quad X(x : A) \Rightarrow \gamma : C & \end{array}$$

(54) $x : A \Rightarrow x : A$ [id]

$$\begin{array}{llll} X \Rightarrow \lambda y\alpha : A \phi B & [\phi R] & X([z : A \phi B\ Y]) \Rightarrow \gamma[z\ \text{'}\ \beta/x] : C & [\phi L] \\ \quad [X\ y : B] \Rightarrow \alpha : A & & \quad Y \Rightarrow \beta : B & \\ & & \quad X(x : A) \Rightarrow \gamma : C & \end{array}$$

$$\begin{array}{llll} X \Rightarrow \lambda y\alpha : B \natural A & [\natural R] & X([Y\ z : B \natural A]) \Rightarrow \gamma[z\ \text{'}\ \beta/x] : C & [\natural L] \\ \quad [y : B\ X] \Rightarrow \alpha : A & & \quad Y \Rightarrow \beta : B & \\ & & \quad X(x : A) \Rightarrow \gamma : C & \end{array}$$

The functional semantics expressed in terms of the lambda calculus is given in e.g. van Benthem (1983) and Moortgat (1988). It is an instance of the Curry-Howard correspondence between constructive implicational logic and the lambda calculus, embodied by an isomorphism between natural deductions and typed lambda terms on a formulas-as-types analogy where beta-reduction corresponds to proof-normalisation:

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

(56)
$$\dfrac{\dfrac{\begin{matrix}\overline{B}^{1}\\ \vdots \\ A\end{matrix}}{B \to A}\,\mathrm{I}_1 \qquad \begin{matrix}\vdots\\ B\end{matrix}}{A}\,\mathrm{E} \qquad \rhd \qquad \begin{matrix}\vdots\\ B\\ \vdots\\ A\end{matrix}$$

4.2 Categorial Logic

Categorial grammar can be seen as an implicational logic without structural rules, i.e. one for which the usual structural rules of intuitionistic logic are dropped. The structural rules delete, duplicate and permute assumptions; their invalidity here resides in the attempt to keep the syntactic operation associated with a statement of formation implicit as the concatenation left-to-right of the the forms associated with the types in that statement. As structural rules are removed the premises in configurations come to be seen not as sets, but multisets, lists, and even trees when associativity is not assumed. (Moortgat 1988, Dosen 1989, Morrill, Leslie, Hepple and Barry 1990).

Once categorial grammar is seen in this light, it is natural to consider addition of other logical type-constructors to obtain a categorial logic with greater expressivity; see e.g. Morrill (1990a) which proceeds with an eye towards linear logic and the Curry-Howard isomorphism for other logical types (Girard, Lafont and Taylor 1989). We must content ourselves here with a terse survey of type-constructors, several of which are used in the next section, presented in a form for which Cut is meant to be eliminable.

Lambek (1958, 1961) contained already product types:

(57)
$$\begin{array}{llll} X\ Y \Rightarrow \alpha,\beta : A \cdot B & [\cdot R] & X(z : A \cdot B) \Rightarrow \gamma[\pi_1 z/x][\pi_2 z/y] : C & [\cdot L] \\ \quad X \Rightarrow \alpha : A & & \quad X(x : A\ y : B) \Rightarrow \gamma : C & \\ \quad Y \Rightarrow \beta : B & & & \end{array}$$

$$\begin{array}{llll} [X\ Y] \Rightarrow \alpha,\beta : A \odot B & [\odot R] & X(z : A \odot B) \Rightarrow \gamma[\pi_1 z/x][\pi_2 z/y] : C & [\odot L] \\ \quad X \Rightarrow \alpha : A & & \quad X([x : A\ y : B]) \Rightarrow \gamma : C & \\ \quad Y \Rightarrow \beta : B & & & \end{array}$$

Here the semantic operations are pairing and projection, as noted by e.g. van Benthem 1987:

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

Derivations can be represented by derivation structures as follows:

(59)
$$\dfrac{\begin{matrix}\vdots\\ A\end{matrix} \quad \begin{matrix}\vdots\\ B\end{matrix}}{A \cdot B}\,\mathrm{I} \qquad\qquad \dfrac{\begin{matrix}\vdots\\ A \cdot B\end{matrix}}{A \qquad B}\,\mathrm{E}$$

To obtain binary infixation we may consider a type-constructor which is a simultaneous forward/backward implication:[5]

[5] Note that the associative $B\backslash\underline{A}/C$ stands in a relation of mutual derivability with $(B\backslash A)/C$ and $B\backslash(A/C)$.

$$\text{(60)}\quad \begin{array}{ll} X \Rightarrow \lambda yz\alpha : B\backslash\!\underline{A}/C & [\backslash\!/\mathrm{R}] \\ \quad y : B\ X\ z : C \Rightarrow \alpha : A & \end{array} \qquad \begin{array}{ll} X(Y\ z : B\backslash\!\underline{A}/C\ Z) \Rightarrow \delta[z\ '\ \beta\ '\ \gamma/x] : D & [\backslash\!/\mathrm{L}] \\ \quad Y \Rightarrow \beta : B & \\ \quad Z \Rightarrow \gamma : C & \\ \quad X(x : A) \Rightarrow \delta : D & \end{array}$$

$$\begin{array}{ll} X \Rightarrow \lambda yz\alpha : B\wr\underline{A}\wr C & [\wr\wr\mathrm{R}] \\ \quad [y : B\ X\ z : C] \Rightarrow \alpha : A & \end{array} \qquad \begin{array}{ll} X([Y\ z : B\wr\underline{A}\wr C\ Z]) \Rightarrow \delta[z\ '\ \beta\ '\ \gamma/x] : D & [\wr\wr\mathrm{L}] \\ \quad Y \Rightarrow \beta : B & \\ \quad Z \Rightarrow \gamma : C & \\ \quad X(x : A) \Rightarrow \delta : D & \end{array}$$

Conjunctive and disjunctive types $A \wedge B$ and $A \vee B$ will pick out expressions which are of both types A and B, and either type A or B, respectively. The rules of inference for $\wedge$ are thus:

$$\text{(61)}\quad \begin{array}{ll} X \Rightarrow \alpha * \beta : A \wedge B & [\wedge R] \\ \quad X \Rightarrow \alpha : A & \\ \quad X \Rightarrow \beta : B & \end{array}$$

$$\begin{array}{ll} X(z : A \wedge B) \Rightarrow \delta[z_1/x] : D & [\wedge\mathrm{L}_a] \\ \quad X(x : A) \Rightarrow \delta : D & \end{array} \qquad \begin{array}{ll} X(z : A \wedge B) \Rightarrow \delta[z_2/y] : D & [\wedge\mathrm{L}_b] \\ \quad X(y : B) \Rightarrow \delta : D & \end{array}$$

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, as was the case for product.

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

The inferences may be represented in derivation structures as follows; $\overline{X}$ signifies cancellation of the sequence X of leaf types.

$$\text{(63)}\quad \dfrac{\begin{array}{cc} X & \overline{X}^{\,n} \\ \vdots & \vdots \\ A & B \end{array}}{A \wedge B}\wedge\mathrm{I}_n \qquad \dfrac{\begin{array}{c} \vdots \\ A \wedge B \end{array}}{A}\wedge\mathrm{E}_a \qquad \dfrac{\begin{array}{c} \vdots \\ A \wedge B \end{array}}{B}\wedge\mathrm{E}_b$$

The rules of inference for $\vee$ are thus:

$$\text{(64)}\quad \begin{array}{ll} X \Rightarrow i\alpha : A \vee B & [\vee\mathrm{R}_a] \\ \quad X \Rightarrow \alpha : A & \end{array} \qquad \begin{array}{ll} X \Rightarrow j\beta : A \vee B & [\vee\mathrm{R}_b] \\ \quad X \Rightarrow \beta : B & \end{array}$$

$$\begin{array}{ll} X(w : A \vee B) \Rightarrow w \rightarrow x.\gamma^i; y.\gamma^j : C & [\vee\mathrm{L}] \\ \quad X(x : A) \Rightarrow \gamma^i : C & \\ \quad X(y : B) \Rightarrow \gamma^j : C & \end{array}$$

Intuitively the operator in (64) is a case operator keyed on i and j. Depending on the labelling, the first or second branch is taken:

$$\text{(65)}\quad \begin{array}{l} i\alpha \rightarrow x.\gamma^i; y.\gamma^j = \gamma^i[\alpha/x] \\ j\alpha \rightarrow x.\gamma^i; y.\gamma^j = \gamma^j[\alpha/y] \end{array}$$

$$\text{(66)}\quad \dfrac{\begin{array}{c} \vdots \\ A \end{array}}{A \vee B}\vee\mathrm{I}_a \qquad \dfrac{\begin{array}{c} \vdots \\ B \end{array}}{A \vee B}\vee\mathrm{I}_b \qquad X \quad \dfrac{\begin{array}{c} \dfrac{\begin{array}{c} \vdots \\ A \vee B \end{array}}{A\ Y X\ B}\vee\mathrm{E}_n \\ \begin{array}{cc} \vdots & \vdots \\ C & C \end{array} \end{array}}{C}n \quad Y$$

Morrill (1989, 1990b) presents modal logic for intensional types. Note first that Montague's theory of types IL proceeds thus: if τ_1, τ_2 are types, then $(\tau_1, \tau_2), \langle s, \tau_1 \rangle$ are types, i.e. indices are not a basic type (as in Gallin's Ty2), rather $\langle s, . \rangle$ is a one-place type-constructor. We use a unary operator, notated $\Box$ for reasons that will become apparent, and give first a rule of 'compositionality for intensions':

(67) $\frac{A_1 \ldots A_n \Rightarrow B}{\Box A_1 \ldots \Box A_n \Rightarrow \Box B}$

This states that where expressions of types $A_1, \ldots, A_n$ can combine to form one in B, expressions of the corresponding intensional types can combine to form one in $\Box B$. The semantic operation involves abstraction over the values of the component expressions across indices; in a Ty2 notation:

$$\text{(68)}\quad \begin{array}{l} y_1 : \Box A_1 \ldots y_n : \Box A_n \Rightarrow \lambda i\gamma[y_1\ '\ i/x_1] \ldots [y_n\ '\ i/x_n] : \Box C \\ \quad x_1 : A_1 \ldots x_n : A_n \Rightarrow \gamma : C \end{array}$$

This modal logic K is common ground between the proposals of van Benthem (1986, appendix), Prijatelj (1989) and Morrill (1989). But for Montague intensions determine extensions and also intensions of intensions: these are constant functions. Thus Morrill (1990b) includes the following:

(69) a. $\Box A \Rightarrow A$

b. $\Box A \Rightarrow \Box\Box A$

The result is the modal logic S4. Where $\Box$X is used to represent configurations with only modal types, the refinement formulation with IL notation is:

$$\text{(70)}\quad \begin{array}{ll} \Box X \Rightarrow {}^{\wedge}\alpha : \Box A & [\Box\mathrm{R}] \\ \quad \Box X \Rightarrow \alpha : A & \end{array} \qquad \begin{array}{ll} X(y : \Box A) \Rightarrow \gamma[{}^{\vee}y/x] : C & [\Box\mathrm{E}] \\ \quad X(x : A) \Rightarrow \gamma : C & \end{array}$$

(71) ${}^{\vee\wedge}\alpha = \alpha$

$$\text{(72)}\quad \dfrac{\begin{array}{c} \vdots \\ A \end{array}}{\Box A}\Box\mathrm{I}^* \qquad \dfrac{\begin{array}{c} \vdots \\ \Box A \end{array}}{X}\Box\mathrm{E}$$

The asterisk on the rule of $\Box$ introduction signifies a condition, in this case that every path from root to leaf contains a modal type not dependent on a discharged assumption.

Morrill, Leslie, Hepple and Barry (1990) discusses structural operators, licensing the structural rules which no longer apply freely. This strategy of building back the structural operations in a *controlled* manner was initiated in Linear Logic (Girard 1987) and the modal nature of these operators is discussed in Dosen (1989). In particular there may be bidirectional or unidirectional S4 permutors:

(73) $\rhd X \Rightarrow \alpha : \rhd A$ [$\rhd$R] $X(x : \rhd A) \Rightarrow \beta : B$ [$\rhd$L]

$\qquad \rhd X \Rightarrow \alpha : A \qquad\qquad X(x : A) \Rightarrow \beta : B$

$X(y : \rhd A\ z : C) \Rightarrow \beta : B$ [$\rhd$P]

$\qquad X(z : C\ y : \rhd A) \Rightarrow \beta : B$

(74)
$$\frac{\begin{matrix}\vdots\\ A\end{matrix}}{\rhd A}\rhd\mathrm{I}^{*} \qquad \frac{\begin{matrix}\vdots\\ \rhd A\end{matrix}}{A}\rhd\mathrm{E} \qquad \frac{\begin{matrix}\vdots\\ \rhd A\end{matrix} \quad \begin{matrix}\vdots\\ D\end{matrix}}{D \quad \rhd A}\rhd\mathrm{P}$$

Inferences for first-order quantification are as follows; semantic interpretation here is taken as identity (the formulas-as-types interpretation is given in Morrill 1990a).

(75) $X \Rightarrow \alpha : \forall vA$ [$\forall$R] $X(x : \forall vA) \Rightarrow \beta : B$ [$\forall$L]

$\qquad X \Rightarrow \alpha : A \qquad\qquad X(x : A[t/v]) \Rightarrow \beta : B$

(v not free in context) (t an individual term)

$X \Rightarrow \alpha : \exists vA$ [$\exists$R] $X(x : \exists vA) \Rightarrow \beta : B$ [$\exists$L]

$\qquad X \Rightarrow \alpha : A[t/v] \qquad\qquad X(x : A) \Rightarrow \beta : B$

(t an individual term) (v not free in context)

(76)
$$\frac{\begin{matrix}\vdots\\ A\end{matrix}}{\forall vA}\forall\mathrm{I}^{**} \qquad \frac{\begin{matrix}\vdots\\ \forall vA\end{matrix}}{A[t/v]}\forall\mathrm{E} \qquad \frac{\begin{matrix}\vdots\\ A[t/v]\end{matrix}}{\exists vA}\exists\mathrm{I} \qquad \begin{matrix}\dfrac{\begin{matrix}\vdots\\ \exists vA\end{matrix}}{A}\exists\mathrm{E}^{*}_{n}\\ \vdots\\ \dfrac{C}{C}n\end{matrix}$$

4.3 Agreement

It is well known that analysing basic type symbols into predicate-argument structures creates a sensitivity and generality in type classification with respect to e.g. morphosyntactic features. Indeed this has inspired approaches which can be seen as type theories formalised entirely by a complex domain of sorted individuals (Pollard and Sag 1989). Typically inclusion of predicational types is represented by placing individual variables at underspecified positions and using unification to test substitutions. Suppose we parameterise NP and CN by an argument for agreement which may take values plural, masculine singular, and feminine singular: pl, sg(m) and sg(f), where sg is a function symbol:

(77)		
	John	:= NP(sg(m))
	Mary	:= NP(sg(f))
	man	:= CN(sg(m))
	men	:= CN(pl)
	woman	:= CN(sg(f))
	women	:= CN(pl)
	this	:= NP(sg(g))/CN(sg(g))
	these	:= NP(pl)/CN(pl)
	the	:= NP(a)/CN(a)
	walk	:= NP(pl)\S
	walks	:= NP(sg(g))\S
	like	:= (NP(pl)\S)/NP(a)
	likes	:= (NP(sg(g))\S)/NP(a)

It is also well known that problems arise in relation to variable scope, and disjunction which necessites some copying in type inference for such 'quantifier free' types. Morrill (1990a) suggests that explicit quantification may be used.

(78) *this* := $\forall g_{\mathrm{gen}}(\mathrm{NP}(\mathrm{sg}(g))/\mathrm{CN}(\mathrm{sg}(g)))$

the := $\forall a_{\mathrm{agr}}(\mathrm{NP}(a)/\mathrm{CN}(a))$

walks := $\exists g_{\mathrm{gen}}\mathrm{NP}(\mathrm{sg}(g))\backslash\mathrm{S}$ or $\forall g_{\mathrm{gen}}(\mathrm{NP}(\mathrm{sg}(g))\backslash\mathrm{S})$

likes := $(\exists g_{\mathrm{gen}}\mathrm{NP}(\mathrm{sg}(g))\backslash\mathrm{S})/\exists a_{\mathrm{agr}}\mathrm{NP}(a)$

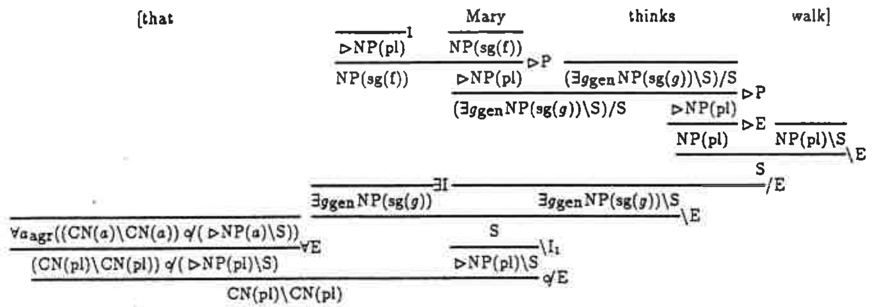
(79)
$$\begin{array}{ccc}\text{the} & \text{man} & \text{walks}\end{array}$$
$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{}{\forall a_{\mathrm{agr}}(\mathrm{NP}(a)/\mathrm{CN}(a))}}{\mathrm{NP}(\mathrm{sg}(\mathrm{m}))/\mathrm{CN}(\mathrm{sg}(\mathrm{m}))}\forall\mathrm{E} \quad \cfrac{}{\mathrm{CN}(\mathrm{sg}(\mathrm{m}))}}{\mathrm{NP}(\mathrm{sg}(\mathrm{m}))}/\mathrm{E}}{\exists g_{\mathrm{gen}}\mathrm{NP}(\mathrm{sg}(g))}\exists\mathrm{I} \quad \cfrac{}{\exists g_{\mathrm{gen}}\mathrm{NP}(\mathrm{sg}(g))\backslash\mathrm{S}}}{\mathrm{S}}\backslash\mathrm{E}$$

4.4 Relativisation

Assigning types (CN\CN)/(NP\S) and (CN\CN)/(S/NP) to the relative pronoun *that* gives only peripheral extraction (although the latter correctly allows relativisation from embedded clauses); instead we can use a type of the kind (CN\CN)/($\rhd$NP\S). Encoding also agreement, and the non-associativity which achieves the CNPC as in section 2, we have the following:

(80) *that* $-\ \lambda x\lambda y\lambda z((x\ 'z) \wedge (y\ 'z))$:= $\forall a_{\mathrm{agr}}((\mathrm{CN}(a)\backslash\mathrm{CN}(a))\not\phi(\rhd\mathrm{NP}(a)\backslash\mathrm{S}))$

(81)



The use of a partially associative categorial grammar for capture of constraints on extraction is first suggested in Oehrle and Zhang (1989); their concern is with non-extractable positions, as distinct from island boundaries however, and their formalisation is different than that here.

4.5 Quantification -

A proposal for the categorial treatment of quantification by means of second-order polymorphism is provided in Emms (1990). There are architectural and computational complications; an approach more immediately in the spirit of categorial logic is provided by Moortgat (1990) proposes to treat quantification by means of an exponentiation type-constructor as follows:

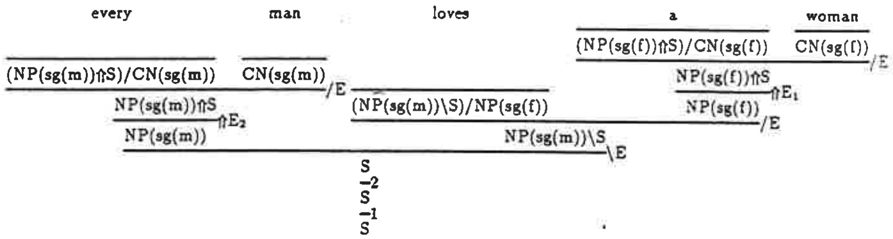
(82) $$\frac{X \Rightarrow \alpha : A}{X \Rightarrow \lambda x(x\,'\,\alpha) : A \Uparrow B}\,[\Uparrow\mathrm{R}] \qquad \frac{Y\ x : A\ Z \Rightarrow \beta : B \qquad X(y : B) \Rightarrow \delta : D}{X(Y\ z : A \Uparrow B\ Z) \Rightarrow \delta[z\,'\,\lambda x\beta/y] : D}\,[\Uparrow\mathrm{L}]$$

(83) $$\frac{\begin{matrix}\vdots\\ A\end{matrix}}{A \Uparrow B}\Uparrow\mathrm{I} \qquad \frac{\begin{matrix}\vdots\\ A \Uparrow B\end{matrix}}{\begin{matrix}A\\ \vdots\\ \dfrac{B}{B}{}^{n}\end{matrix}}\Uparrow\mathrm{E}_n$$

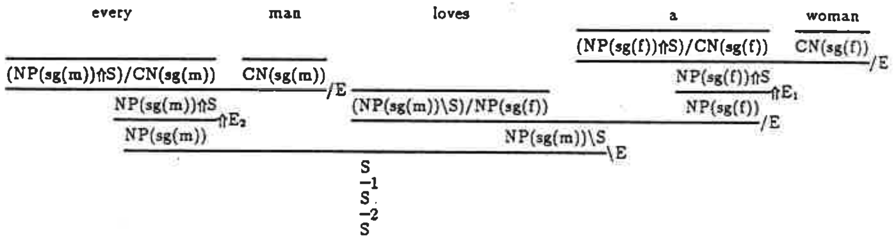
(84) $$\begin{array}{lll} a & -\ \lambda x\lambda y\exists z((x\,'\,z) \wedge (y\,'\,z)) & := \forall g_{\mathrm{gen}}((\mathrm{NP}(sg(g))\Uparrow\mathrm{S})/\mathrm{CN}(sg(g))) \\ every & -\ \lambda x\lambda y\forall z((x\,'\,z) \rightarrow (y\,'\,z)) & := \forall g_{\mathrm{gen}}((\mathrm{NP}(sg(g))\Uparrow\mathrm{S})/\mathrm{CN}(sg(g))) \end{array}$$

The following two derivations, slightly abbreviated with regards to agreement, give the expected readings.

(85)



(86)



In general, quantifiers will be able to take scope at any dominating sentence level, but the interaction of ⇑L and the non-associative functor type for relative pronouns blocks quantification out of relative clauses, as required.[6]

4.6 Reflexivisation

The phenomena of relativisation and quantification are long-distance, and the grammar developed so far characterises this. If we attempt to treat reflexivisation in an analogous manner, by assignment of type ((NP\S)/NP)\(NP\S) (or NP⇑(NP\S)) the incorrect prediction is made that this also is long-distance. The problem comes down to an absence of sensitivity in type classification to clause boundaries; thus e.g. (NP\S)/NP includes both verb phrases lacking main clause objects, and embedded clause objects. Morrill (1989, 1990b) observes that once types are intensional, this information becomes visible in the language of types when each clause is assumed to form an intensional domain. Thus intensionality provides a term of reference for locality.[7]

Intensional type assignments are as follows:[8]

[6] Montague's PTQ, incidently, fails to capture the scope-islandhood of relative clauses: quantifying-in applies freely.

[7] We could proceed by introducing modal operators tuned exclusively to boundedness facts, as in Hepple (1990b), but before such a move is made it seems interesting to see the extent to which the facts can be managed by devices independently motivated on semantic grounds.

[8] That certain meanings are logical or that proper names might be rigid designators is not taken to indicate that lexical types should be non-modal, since the information as to the extension across indices must be available for any element to appear compositionally in an intensional context, even if the types cannot capture a generalisation such as that the extension is the same across indices.

(87) *John* – ˆjohn := □NP
Mary – ˆmary := □NP
walks – walk := □(NP\S)
man – man := □CN
the – the := □(NP/CN)
thinks – think := □((NP\S)/□S)
likes – like := □((NP\S)/NP)

Against the basic types of (87), the following type assignments now capture the respective boundedness of reflexivisation on the one hand, and unboundedness of relativisation and quantification on the other.

(88) *himself* — $^\wedge\lambda x\lambda y((x\,'\,y)\,'\,y)$:= □(NP⇑(NP\S))
a — $^\wedge\lambda x\lambda y\exists z((x\,'\,z) \wedge (y\,'\,^\wedge z))$:= □((□NP)⇑S)/CN)
every — $^\wedge\lambda x\lambda y\forall z((x\,'\,z) \wedge (y\,'\,^\wedge z))$:= □((□NP)⇑S)/CN)
that — $^\wedge\lambda x\lambda y\lambda z((x\,'\,^\wedge z) \wedge (y\,'\,z))$:= $\Box((\mathrm{CN}\backslash\mathrm{CN})\not/(\triangleright\Box\mathrm{NP}\backslash\mathrm{S}))$

4.7 Coordination

The primary empirical motivation for the generous theories of formation employed in categorial grammar is drawn from coordination (see e.g. Steedman 1985, 1987, Dowty 1988, Morrill 1988, Barry and Pickering 1990). Thus if, as seems reasonable, conjuncts have a unitary meaning, then non-constituent coordination (right node raising, left node raising, and so on) indicates that many 'non-constituents' must be defined as meaningful in the same way as traditional constituents. The question as to which kinds of expressions can form constituents, and what kinds of type-constructors would capture the generalisations, is too extensive to be addressed here; rather, types for some basic constituent coordinators are listed using the present machinary.

A well-noted constraint on the interaction of relativisation and coordination is that there can be no relativisation out of a coordinate structure unless is is across-the-board, i.e. from each conjunct: the Coordinate Structure Constraint minus Across-The-Board Exception (CSC–ATBE).

(89) a. *the man that walks and John sings
b. *the man that John sings and walks
c. the man that walks and sings

(90) a. *the man that John likes and Mary sings
b. *the man that John sings and Mary likes
c. the man that John likes and Mary loves

However, assignment of type $\mathrm{S}\backslash\underline{\mathrm{S}}/\mathrm{S}$ (or (S\S)/S or S\(S/S)) to a coordinator allows CSC–ATBE violation:

(91) that
$$\dfrac{\dfrac{\dfrac{\overline{\mathrm{NP}}^{1}\quad\dfrac{\text{walks}}{\mathrm{NP}\backslash\mathrm{S}}}{\mathrm{S}}\backslash E\quad\dfrac{\text{and}}{\mathrm{S}\backslash\underline{\mathrm{S}}/\mathrm{S}}\quad\dfrac{\dfrac{\text{John}}{\mathrm{NP}}\quad\dfrac{\text{sings}}{\mathrm{NP}\backslash\mathrm{S}}}{\mathrm{S}}\backslash E}{\mathrm{S}}\backslash\!/E}{\mathrm{NP}\backslash\mathrm{S}}\backslash I^{1}$$

Just as the island status of relative clauses was achieved by making relative pronouns non-associative functors, the CSC is achieved by using $\mathrm{S}\not\backslash\underline{\mathrm{S}}\not/\mathrm{S}$. Then for ATBE:

(92) *and* – $\lambda xy\lambda z((x\,'\,z) \wedge (y\,'\,z))$:= $(\triangleright\mathrm{NP}\backslash\mathrm{S})\not\backslash\underline{\triangleright\mathrm{NP}\backslash\mathrm{S}}\not/(\triangleright\mathrm{NP}\backslash\mathrm{S})$

In the relativisation case the non-associativity prevented also quantification out of relative clauses; similarly here it correctly prevents the quantification out of coordinate structures according to which would exist for following:[9]

(93)*A man walks or John sings – $\exists x((\mathbf{man}\,'\,x) \wedge (\ldots \vee \ldots))$

For verb phrase conjunction we can have:

(94) *and* – $\lambda xy\lambda z((x\,'\,z) \wedge (y\,'\,z))$:= $(\mathrm{NP}\backslash\mathrm{S})\not\backslash\underline{\mathrm{NP}\backslash\mathrm{S}}\not/(\mathrm{NP}\backslash\mathrm{S})$

This correctly prevents the wide scope conjunction in (95), which is the classic pitfall of a deletion analysis.

(95)*A man walks and sings – $\exists x \ldots \wedge \exists x \ldots$

But what about (96)?

(96) Every woman walks or sings – $\forall x \ldots \vee \forall x \ldots$

Partee and Rooth (1983) claim no wide scope "or" reading, but my judgement, in accordance with Hendriks (1990), is that it exists. More clear is that (97) *does* have a reading in which disjunction takes scope over the embedding verb.

(97) John claims every woman walks or sings

Accordingly, it will be assumed that a type is required for a verb phrase disjunctive structure which enables the coordinator to take scope at any superordinate sentence level. Moortgat's exponentiator achieves this:

(98) *or* – $\lambda xy\lambda z((z\,'\,x) \vee (z\,'\,y))$:= $(\mathrm{NP}\backslash\mathrm{S})\not\backslash\underline{(\mathrm{NP}\backslash\mathrm{S})\Uparrow S}\not/(\mathrm{NP}\backslash\mathrm{S})$

Quantified and definite noun phrases can both be coordinated and can be coordinated with each other. A shared type is S/(NP\S) since NP ⇒ S/(NP\S) and NP⇑S ⇒ S/(NP\S). Subject coordination can be based on this type, but there are complications with agreement. For conjunction, the generalisation suggests itself that conjoining subjects of any agreement yields a plural coordinate structure:[10]

[9] Montague's PTQ fails to capture this constraint, as noted by Hendriks 1990 who cites Paul Dekker p.c.

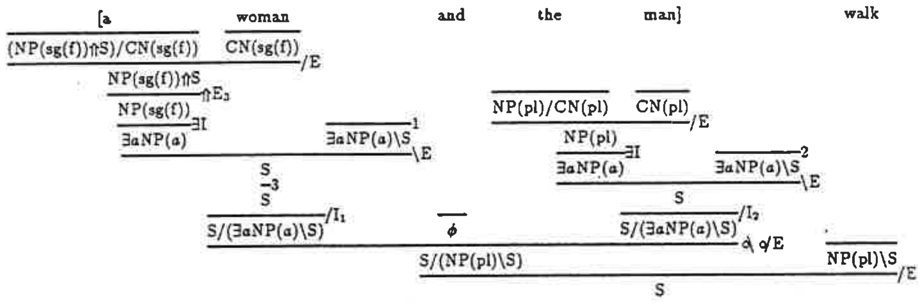
[10] This cannot be the whole story: compare *every man and every boy walks/*walk* and *a man and a boy walk/*walks*. Since we currently make no distinction between the types of these two quantifiers, no handle is available here.

(99) Mary/the man/the women and John/the men/the woman walk/like themselves/*walks

Then the lexical assignment for subject conjunction is thus:

(100) $\mathit{and}\quad - \lambda xy\lambda z((x\ '\ z) \wedge (y\ '\ z)) \quad := \phi$
where $\phi = (\mathrm{S}/(\exists a\mathrm{NP}(a)\backslash S))\backslash\underline{\mathrm{S}/(\mathrm{NP(pl)}\backslash S)}/(\mathrm{S}/(\exists a\mathrm{NP}(a)\backslash S))$

(101)



For disjunction, it seems possible to say that subject disjuncts must be forms individually able to reach agreement with the verb phrase, but which need not have identical agreement with each other:

(102) a. Mary or Suzy walks/likes herself/*walk

b. John or Mary walks/*walk

c. the men or the women walk/like themselves/*walks

(103) $\mathit{or}\quad - \lambda xy\lambda z((z\ '\ ix) \vee (z\ '\ jy))$
$:= \forall a\forall a'((\mathrm{S}/(\mathrm{NP}(a)\backslash S))\backslash\underline{((\mathrm{S}/(\mathrm{NP}(a)\backslash S)) \vee (\mathrm{S}/(\mathrm{NP}(a')\backslash S))) \Uparrow S}/(\mathrm{S}/(\mathrm{NP}(a')\backslash S)))$

5 Prolog Implementation and Illustrative Log

The basic idea of Prolog implementation for Cut-free ACG derivation search is discussed at length in Moortgat (1988). Completeness relies on the correctness of the assumption that Cut is eliminable, with respect to semantic as well as syntactic interpretation. For the fairly extensive categorial logic employed here proof would be a considerable task. Within the space of Cut-free derivations there is still a great deal of derivational equivalence; note especially that decomposing independent slash types on the left in different orders, and decomposing modal types on the left in different orders, are both factorial in their options. König (1989) and Hepple (1990b) develop a notion of normal Cut-free ACG derivation which involves a commitment to try decomposing types in certain orders; this method is employed here and extended also to cover modals. Completeness of the implementation further rests on the assumption that these normal derivations are fully representative. A final spead-up is obtained by using also a generalisation of the notion of reachability in König (1989).

Prolog metavariables are used to represent individual variables in types, and variables in lambda terms, and there are potential problems with scope and proof search requiring different instantiations. The ∀L and ∃R backward inferences are carried out by substituting new Prolog variables for the quantified variables, ∀R and ∃L by substituting new Prolog constants. The procedure `substitute/4` does not actually instanciate the quantified variables, so that alternative substitutions do not fail through an attempt to instanciate the saem Prolog variable to two different values. This coding is due to Guy Barry. The same device can be used for lambda reduction, but here use of metavariables for object variables causes another problem: in recognising redexes for normalisation, a variable would match any redex scheme. Therefore `eval/2` invokes `numbervars/3` to freeze terms, and normalises via explicit substitution of terms for constants.[11]

Sentences 1–4 illustrate simple sentence structures; note that in the intensional translations names appear as constants standing for the individual they are assumed to rigidly designate. Proper names and definite noun phrases are not treated as scoping elements, though they could easily be given the same kind of Montague 'quantifying in' treatment as other quantifier terms. 5–12 show agreement between demonstrative articles, subject noun, and verb. Examples 13–16 show that the required agreement between a matrix subject relative clause and its antecedent is enforced; 17 shows object relativisation; 18–21 show the agreement for an embedded subject relative clause, and 22 demonstrates the long-distance nature of relativisation. Example 23 shows the islandhood of relative clauses with respect to further left extraction.

Examples 24 and 25 show quantification, 26, the classic quantificational ambiguity. There are three readings for 27: one with the existential construed at the lower clause, and two others with the quantifiers alternating in scope at the matrix level. In a similar way, 28 has eight readings. Examples 29 and 30 demonstrate synonomy arising in the interaction of relativisation and existential quantification. Example 31 shows that relative clauses are ceilings to quantification as well as islands to relativisation: there is no reading with the universal taking wide scope.

32-35 show singular reflexive agreement for gender; 36 shows boundedness of reflexivisation. 37–40 show corresponding effects for plurals.

Example 41–45 show coordination, including singularity of a singular subject disjunctions, and plurality of singular subject conjunctions. 46 has two readings, for matrix and embedded coordination; 47 has the additional wide-scope 'or' reading where the embedded disjunction is construed at the matrix level. 48 and 49 illustrate the same effect, this time for embedded subject coordination as opposed to embedded verb phrase coordination.

Examples 50 and 51 show that relative clauses block wide scope of 'or' as well as 'and'; 52–57 demonstrate the ATB–CSC. 58 shows coordination of a quantified and a non-quantified subject, and that the scope of the quantifier is correctly kept within that of the coordinator. In 59, the quantifier outside the coordinate structure is correctly prevented from taking scope within the conjunction; in 60 the wide scope and narrow scope coordinations have the same logical weight; in 62 there is the controversial wide scope 'or'. 63 has one reading with the coordination at the matrix level; with the coordination subordinate, the disjunction may be at either sentence level; 64 shows several readings arising from the interaction of an additional

[11] Note that neither of these two ways of 'getting away' with metalanguage variables for object language ones would capture the blocking of scope by a new local binding of the same variable; luckily this situation does nor arise here.

quantifier. Example 65 delivers two readings for disjoined indefinites in an embedded clause; 66–73 show agreement in interaction of reflexivisation and subject coordination.

```
Script started on Thu Dec 20 15:53:59 1990
ruulo2% cat cg8.pl

% Get term simplifier

:- [eval].

:- dynamic(symb_rec/1).

% Assignment and lexical assignment

:- op(500,xfx,:).
:- op(500,xfx,':=').

% Form-meaning link

:- op(450,xfx,-).

% Implications

:- op(400,xfx,/).
:- op(400,xfx,\).

% Conjunction and disjunction

:- op(400,xfx,&).
:- op(400,xfx,+).

% Intensionality modal

:- op(400,fx,#).

% Exponentiator

:- op(400,fx,^).

% Permutor

:- op(400,fx,>).

% Quantifiers

:- op(400,xfx,all).
:- op(400,xfx,xst).
```

```
% top(+Words,-X,+A) means that Words parses with meaning
% X of type A

top(Words,X,A) :-
        setof(Y,prove1(Words,Y:A),Ys),
        member(X,Ys).

prove1(Words,X:A) :-
        premises(Words,Config),
        prove(Config,Y:A),
        numbervars(Y,0,_),
        eval(Y,X).

% premises(+Words,-Config) means that Words corresponds
% to configured type assignment statements Config

premises(Words,Config) :-
        premises(Config,□,Words).

premises(Config,Config,□).

premises(Result,Config,[Word|Words]) :-
        Word - X := A,
        append(Config,[X:A],ConfigA),
        premises(Result,ConfigA,Words).

% Element needing to be leftmost in brackets (relative
% pronoun)

premises(Result,Config,[Word|Words1Words2]) :-
        lb(Word - X := A),
        append(Words1,Words2,Words1Words2),
        \+(Words1 = □),
        premises(Words1,Config1),
        append(Config,[[X:A|Config1]],ConfigAConfig1),
        premises(Result,ConfigAConfig1,Words2).

% Element needing to be in brackets (coordinator)

premises(Result,Config1Config2,[Word|Words3Words4]) :-
        b(Word - X := A),
        append(Words3,Words4,Words3Words4),
        \+(Words3 = □),
        premises(Words3,Config3),
        append(Config1,Config2,Config1Config2),
        \+(Config2 = □),
        append(Config2,[X:A|Config3],Config2AConfig3),
        append(Config1,[Config2AConfig3],Config1Config2AConfig3),
        premises(Result,Config1Config2AConfig3,Words4).

% subconfig(Config,RemConfig,P,SubConfig) means that RemConfig is the
% result of replacing the non-trivial, but possibly improper,
% subconfiguration SubConfig in Config by P

subconfig(Gamma,P,P,Gamma) :-
        \+(Gamma = □).

subconfig(XYZ,XWZ,P,V) :-
        append(X,[Y|Z],XYZ),
        subconfig(Y,W,P,V),
        append(X,[W|Z],XWZ).

% Right rules

% \R

prove(Gamma,[lmd,A,B]:X\Y) :-
        prove([A:X|Gamma],B:Y).

% /R

prove(Gamma,[lmd,A,B]:Y/X) :-
        append(Gamma,[A:X],GammaAX),
        prove(GammaAX,B:Y).

% &R

prove(Gamma,[pair,A,B]:X&Y) :-
        prove(Gamma,A:X),
        prove(Gamma,B:Y).

% +R

prove(Gamma,[i,A]:X+_) :-
        prove(Gamma,A:X).

prove(Gamma,[j,B]:_+Y) :-
        prove(Gamma,B:Y).

% [\]R

prove(Gamma,[lmd,A,B]:[X\Y]) :-
        prove([[A:X|Gamma]],B:Y).
```

```
% [/]R

prove(Gamma,[lmd,A,B]:[Y/X]) :-
        append(Gamma,[A:X],GammaAX),
        prove([GammaAX],B:Y).

% ^R

prove(Gamma,[lmd,A,[app,A,B]]:X^_) :-
        prove(Gamma,B:X).

% allR

prove(Gamma,X:V all A) :-
        gen_symb(U),
        substitute(U,V,A,A1),
        prove(Gamma,X:A1).

% xstR

prove(Gamma,X:V xst A) :-
        substitute(_,V,A,A1),
        prove(Gamma,X:A1).

% aR

prove(Gamma,[lmd2,A,C,B]:a(X,Y,Z)) :-
        append([A:X|Gamma],[C:Z],XGammaZ),
        prove(XGammaZ,B:Y).

% nR

prove(Gamma,[lmd2,A,C,B]:n(X,Y,Z)) :-
        append([A:X|Gamma],[C:Z],XGammaZ),
        prove([XGammaZ],B:Y).

% Switch to left for Konig/Hepple processing

prove(Gamma,X:A) :-
        subconfig(Gamma,LL,P,S1FUS2),
        append(S1,[F:U|S2],S1FUS2),
        reachable(A,U),
        provel(LL,P,S1,F:U,S2,X:A).

provel(L,L,□,A:X,□,A:X) :-
        basic_type(X).
```

```
% \L

provel(LL,P,Gamma1Delta,A:Y\X,Gamma2,CZ) :-
        append(Gamma1,Delta,Gamma1Delta),
        prove(Delta,B:Y),
        provel(LL,P,Gamma1,[app,A,B]:X,Gamma2,CZ).

% /L

provel(LL,P,Gamma1,A:X/Y,DeltaGamma2,CZ) :-
        append(Delta,Gamma2,DeltaGamma2),
        prove(Delta,B:Y),
        provel(LL,P,Gamma1,[app,A,B]:X,Gamma2,CZ).

% &L

provel(LL,P,Gamma,A:X&_,Delta,CZ) :-
        append(Gamma,[[fst,A]:X|Delta],P),
        prove(LL,CZ).

provel(LL,P,Gamma,A:_&Y,Delta,CZ) :-
        append(Gamma,[[snd,A]:Y|Delta],P),
        prove(LL,CZ).

% allL

provel(LL,P,Gamma,X:V all A,Delta,CZ) :-
        substitute(_,V,A,A1),
        provel(LL,P,Gamma,X:A1,Delta,CZ).

% xstL

provel(LL,P,Gamma,X:V xst A,Delta,CZ) :-
        gen_symb(U),
        substitute(U,V,A,A1),
        provel(LL,P,Gamma,X:A1,Delta,CZ).

% +L

provel(LL,P,Gamma,A:X+Y,Delta,[case,A,B,C,D,E]:Z) :-
        provel(LL,P,Gamma,B:X,Delta,C:Z),
        provel(LL,P,Gamma,D:Y,Delta,E:Z).


% [\]L
```

```
provel(LL,[app,A,B]:X,Delta,A:[Y\X],□,CZ) :-
        prove(Delta,B:Y),
        prove(LL,CZ).

% [/]L

provel(LL,[app,A,B]:X,□,A:[X/Y],Delta,CZ) :-
        prove(Delta,B:Y),
        prove(LL,CZ).

% ~L

provel(LL,P,G1D1,A:X~Y,D2G2,CZ) :-
        append(G1,D1,G1D1),
        append(D2,G2,D2G2),
        append(D1,[B:X|D2],D1BD2),
        prove(D1BD2,C:Y),
        append(G1,[[app,A,[lmd,B,C]]:Y|G2],P),
        prove(LL,CZ).

% #R

provel(LL,P,Gamma,A: #X,Delta,[up,B]: #Y) :-
        append(Gamma,[A: #X|Delta],P),
        modal(LL),
        prove(LL,B:Y).

% #L

provel(LL,P,Gamma,A: #X,Delta,CZ) :-
        provel(LL,P,Gamma,[dn,A]:X,Delta,CZ).

% aL

provel(LL,P,G1D1,B:a(X,Y,Z),D2G2,CZ) :-
        append(G1,D1,G1D1),
        append(D2,G2,D2G2),
        prove(D1,A:X),
        prove(D2,C:Z),
        appendn([G1,[[app2,B,A,C]:Y],G2],P),
        prove(LL,CZ).

% nL

provel(LL,[app2,B,A,C]:Y,Gamma,B:n(X,Y,Z),Delta,CZ) :-
        prove(Gamma,A:X),
        prove(Delta,C:Z),
          prove(LL,CZ).

% >R

provel(LL,P,Gamma,X: (>A),Delta,Y: (>B)) :-
          append(Gamma,[X: (>A)|Delta],P),
          rightexmodal(LL),
          prove(LL,Y:B).

% >L

provel(LL,P,Gamma,X:(>A),Delta,CZ) :-
          append(Gamma,[X:A|Delta],P),
          prove(LL,CZ).

% >P

provel(LL,P,Gamma,X:(>A),[YB|Delta],CZ) :-
          append(Gamma,[YB,X:(>A)|Delta],P),
          prove(LL,CZ).

append([],L,L).

append([H|L1],L2,[H|L]) :-
          append(L1,L2,L).

appendn([],[]).

appendn([L1|Ls],L) :-
          append(L1,L2,L),
          appendn(Ls,L2).

member(H,[H|_]).
member(X,[_|T]) :-
          member(X,T).

% Konig reachability

reachable(X,Y) :-
          \+ \+ reachable1(X,Y).

reachable1(X,X).
reachable1(X,Y/_) :-
          reachable1(X,Y).
reachable1(X,_\Y) :-
          reachable1(X,Y).
reachable1(X,[Y/_]) :-
```

```
          reachable1(X,Y).
reachable1(X,[_\Y]) :-
          reachable1(X,Y).
reachable1(X,a(_,Y,_)) :-
          reachable1(X,Y).
reachable1(X,n(_,Y,_)) :-
          reachable1(X,Y).
reachable1(X,#Y) :-
          reachable1(X,Y).
reachable1(#X,Y) :-
          reachable1(X,Y).

reachable1(X,_^Y) :-
          reachable1(X,Y).

reachable1(_,(>_)).

reachable1(>A,B) :-
          reachable1(A,B).

reachable1(C,A+B) :-
          reachable1(C,A),
          reachable1(C,B).

reachable1(C,A&_) :-
          reachable1(C,A).
reachable1(C,_&B) :-
          reachable1(C,B).

reachable1(C,_ all A) :-
          reachable1(C,A).
reachable1(C,_ xst A) :-
          reachable1(C,A).

modal([]).
modal(_: #_).
modal([H|T]) :-
          modal(H),
          modal(T).

rightexmodal([]).
rightexmodal(_:(>_)).
rightexmodal([H|T]) :-
          rightexmodal(H),
          rightexmodal(T).

basic_type(s).
basic_type(sp).
basic_type(np(_)).
basic_type(cn(_)).

symb_rec(0).

gen_symb(N1) :- retract(symb_rec(N)),
        N1 is N+1,
        assert(symb_rec(N1)).

% Barry substitution

substitute(Val,Var,InTerm,OutTerm) :-
        bagof(Term,Var^(Var=Val,Term=InTerm),[OutTerm]).

% lexical entries

a           -  [up,[lmd,P,[lmd,Q,[xst,X,[both,[app,P,X],[app,Q,[up,X]]]]]]]
            := #(G all ((((#np(sg(G)))^s))/cn(sg(G)))).
bill        -  [up,bill]
            := #np(sg(m)).
boy         -  boy
            := #cn(sg(m)).
boys        -  boys
            := #cn(pl).
claim       -  claim
            := #((np(pl)\s)/(#s)).
claims      -  claim
            := #(((G xst np(sg(G)))\s)/(#s)).
every       -  [up,[lmd,P,[lmd,Q,[all,X,[imply,[app,P,X],[app,Q,[up,X]]]]]]]
            := #(G all ((((#np(sg(G)))^s))/cn(sg(G)))).
girl        -  girl
            := #cn(sg(f)).
herself     -  [up,[lmd,X,[lmd,Y,[app,[app,X,Y],Y]]]]
            := #(np(sg(f))^(np(sg(f))\s)).
himself     -  [up,[lmd,X,[lmd,Y,[app,[app,X,Y],Y]]]]
            := #(np(sg(m))^(np(sg(m))\s)).
john        -  [up,john]
            := #np(sg(m)).
like        -  like
            := #((np(pl)\s)/(A xst np(A))).
likes       -  like
            := #(((G xst np(sg(G)))\s)/(A xst np(A))).
loves       -  loves
            := #(((G xst np(sg(G)))\s)/(A xst np(A))).
man         -  man
            := #cn(sg(m)).
```

```
mary        -  [up,mary]
            := #np(sg(f)).
men         -  men
            := #cn(pl).
run         -  run
            := #(np(pl)\s).
runs        -  run
            := #((G xst np(sg(G)))\s).
sing        -  sing
            := #(np(pl)\s).
sings       -  sing
            := #((G xst np(sg(G)))\s).
suzy        -  suzy
            := #np(sg(f)).
the         -  the
            := #(A all (np(A)/cn(A))).
themselves  -  [up,[lmd,X,[lmd,Y,[app,[app,X,Y],Y]]]]
            := #(np(pl)^(np(pl)\s)).
these       -  these
            := #(np(pl)/cn(pl)).
think       -  think
            := #((np(pl)\s)/(#s)).
thinks      -  think
            := #(((G xst np(sg(G)))\s)/(#s)).
this        -  this
            := #(G all (np(sg(G))/cn(sg(G)))).
walk        -  walk
            := #(np(pl)\s).
walks       -  walk
            := #((G xst np(sg(G)))\s).
woman       -  woman
            := #cn(sg(f)).

% Sentence coordination

b(and       -  [up,[lmd2,X,Y,[both,X,Y]]]
            := #n(s,s,s)).

b(or        - [up,[lmd2,X,Y,[either,X,Y]]]
            := #n(s,s,s)).

% Verb phrase coordination

b(and
-  [up,[lmd2,X,Y,[lmd,Z,[both,[app,X,Z],[app,Y,Z]]]]]
:= #(A all n(np(A)\s,np(A)\s,np(A)\s))).
```

```
b(or
-  [up,[lmd2,X,Y,[lmd,Z,[either,[app,Z,[up,X]],[app,Z,[up,Y]]]]]]
:= #(A all n(np(A)\s,#((np(A)\s))^s,np(A)\s))).

% Subject coordination

b(and
-  [up,[lmd2,X,Y,[lmd,Z,[both,[app,X,Z],[app,Y,Z]]]]]
:= #n(s/((A xst np(A))\s),s/(np(pl)\s),s/((A xst np(A))\s))).

b(or
-  [up,[lmd2,X,Y,[lmd,Z,[either,[app,Z,[up,[i,X]]],[app,Z,[up,[j,Y]]]]]]]
:= #(A all (A1 all n(s/(np(A)\s),#(((s/(np(A)\s))+(s/(np(A1)\s))))^s,s/(np(A1)\s))))).

% ATBE

b(and
- [up,[lmd2,X,Y,[lmd,Z,[both,[app,X,Z],[app,Y,Z]]]]]
:= #(A all n((>np(A))\s,(>np(A))\s,(>np(A))\s))).

b(or
- [up,[lmd2,X,Y,[up,[lmd,Z,[either,[app,X,Z],[app,Y,Z]]]]]]
:= #(A all n((>np(A))\s,#((>np(A))\s),(>np(A))\s))).

% Relative pronoun

lb(that
- [up,[lmd,X,[lmd,Y,[lmd,Z,[both,[app,X,[up,Z]],[app,Y,Z]]]]]]
:= (# (A all ([(cn(A)\cn(A))/((>(#np(A)))\s)])))).

% test(N) tests example N

test(N) :-
        str(N,Str,A),
        nl, nl, write(N), tab(2), write(Str),
        write(' => '), write(A), write('?'), nl,
        test1(Str,A).

test1(Str,A) :-
        top(Str,X,A),
        write(X), nl, fail.

str(1,[mary,walks],s).
str(2,[mary,likes,john],s).
str(3,[mary,loves,the,men],s).
str(4,[bill,claims,the,man,thinks,the,woman,walks],s).
```

```
str(5,[this,man,walks],s).
str(6,[this,man,walk],s).
str(7,[this,men,walks],s).
str(8,[this,men,walk],s).
str(9,[these,man,walks],s).
str(10,[these,man,walk],s).
str(11,[these,men,walks],s).
str(12,[these,men,walk],s).

str(13,[the,man,that,walks],np(_)).
str(14,[the,man,that,walk],np(_)).
str(15,[the,men,that,walks],np(_)).
str(16,[the,men,that,walk],np(_)).

str(17,[the,man,that,suzy,likes],np(_)).

str(18,[the,man,that,mary,thinks,walks],np(_)).
str(19,[the,man,that,mary,thinks,walk],np(_)).
str(20,[the,men,that,mary,thinks,walks],np(_)).
str(21,[the,men,that,mary,thinks,walk],np(_)).

str(22,[the,man,that,mary,thinks,bill,claims,suzy,likes],np(_)).

str(23,[the,man,that,mary,likes,the,woman,that,loves],np(_)).

str(24,[a,man,walks],s).
str(25,[john,likes,a,man],s).
str(26,[every,man,loves,a,woman],s).
str(27,[every,man,claims,a,woman,sings],s).
str(28,[john,claims,every,boy,thinks,a,woman,walks],s).

str(29,[a,man,that,walks,sings],s).
str(30,[a,man,that,sings,walks],s).

str(31,[a,man,that,likes,every,woman,sings],s).

str(32,[john,likes,himself],s).
str(33,[john,likes,herself],s).
str(34,[mary,likes,himself],s).
str(35,[mary,likes,herself],s).

str(36,[john,thinks,mary,likes,himself],s).

str(37,[the,boys,like,themselves],s).
str(38,[the,boy,likes,themselves],s).
str(39,[the,boy,like,themselves],s).
str(40,[the,boys,think,mary,likes,themselves],s).
str(41,[john,walks,and,mary,sings],s).
str(42,[john,walks,or,sings],s).
str(43,[john,or,mary,walks],s).
str(44,[john,and,mary,walks],s).
str(45,[john,and,mary,walk],s).

str(46,[mary,thinks,john,walks,and,sings],s).
str(47,[mary,thinks,john,walks,or,sings],s).

str(48,[bill,thinks,john,and,mary,walk],s).
str(49,[bill,thinks,john,or,mary,walks],s).

str(50,[the,man,that,walks,and,runs,sings],s).
str(51,[the,man,that,walks,or,runs,sings],s).

str(52,[the,man,that,john,sings,and,mary,likes],np(sg(m))).
str(53,[the,man,that,john,likes,and,mary,sings],np(sg(m))).
str(54,[the,man,that,john,likes,and,mary,loves],np(sg(m))).
str(55,[the,man,that,john,sings,or,mary,likes],np(sg(m))).
str(56,[the,man,that,john,likes,or,mary,sings],np(sg(m))).
str(57,[the,man,that,john,likes,or,mary,loves],np(sg(m))).

str(58,[a,woman,or,john,walks],s).
str(59,[a,woman,walks,and,sings],s).
str(60,[a,woman,walks,or,sings],s).
str(61,[every,woman,walks,and,sings],s).
str(62,[every,woman,walks,or,sings],s).

str(63,[john,claims,mary,walks,or,sings],s).

str(64,[john,claims,every,woman,walks,or,sings],s).

str(65,[john,thinks,a,man,or,a,woman,sings],s).

str(66,[mary,or,suzy,likes,herself],s).
str(67,[mary,or,suzy,likes,themselves],s).
str(68,[mary,or,suzy,like,herself],s).
str(69,[mary,or,suzy,like,themselves],s).
str(70,[john,and,bill,likes,himself],s).
str(71,[john,and,bill,likes,themselves],s).
str(72,[john,and,bill,like,himself],s).
str(73,[john,and,mary,like,themselves],s).
ruulo2% cat eval.pl

% eval(+Term,-NF) lambda-converts a lambda term Term to its normal
% form NF.  The terms may be freely generated by n-ary operators
```

```
% which form the heads of lists with their list of operands as tail.
% The operators include app which takes two terms as operands, and
% lmd which takes a variable and a term.

eval(X,NF) :-
        numbervars(X,0,_),
        eval1(X,NF).

eval1(Term,NF) :-
        contract(Term, Term1), !,
        eval1(Term1,NF).

eval1(Term,Term).

contract([app,[lmd,A,B],C],D) :-
        subst(C,A,B,D).

contract([app2,[lmd2,A,B,C],D,E],F) :-
        subst(D,A,C,G),
        subst(E,B,G,F).

contract([fst,[pair,A,_]],A).
contract([snd,[pair,_,B]],B).

contract([case,[i,A],B,C,_,_],F) :-
        subst(A,B,C,F).

contract([case,[j,A],_,_,D,E],F) :-
        subst(A,D,E,F).

contract([dn,[up,A]],A).

contract([H|T],[H|T1]) :-
        contractlist(T,T1).

contractlist([T|Ts],[T1|Ts]) :-
        contract(T,T1).
contractlist([T|Ts],[T|Ts1]) :-
        contractlist(Ts,Ts1).

subst(A,B,B,A).

subst(_,_,C,C) :- atom(C).
subst(_,_,C,C) :- C = '$VAR'(_).

subst(A,B,[H|Ts],[H1|Ts1]) :-
        subst(A,B,H,H1),
        subst(A,B,Ts,Ts1).
ruulo2% prolog

Quintus Prolog Release 2.5 (Sun-4, SunOS 4.1)
Copyright (C) 1990, Quintus Computer Systems, Inc.  All rights reserved.
1310 Villa Street, Mountain View, California  (415) 965-7700

| ?- compile(cg8).
[compiling /users.lot/morrill/Dyana/Progs/cg8.pl...]
 [consulting /users.lot/morrill/Dyana/Progs/eval.pl...]
 [eval.pl consulted 0.117 sec 2,244 bytes]
[cg8.pl compiled 7.500 sec 22,400 bytes]

yes
| ?- test(_).


1  [mary,walks] => s?
[app,[dn,walk],mary]


2  [mary,likes,john] => s?
[app,[app,[dn,like],john],mary]


3  [mary,loves,the,men] => s?
[app,[app,[dn,loves],[app,[dn,the],[dn,men]]],mary]


4  [bill,claims,the,man,thinks,the,woman,walks] => s?
[app,[app,[dn,claim],[up,[app,[app,[dn,think],[up,[app,[dn,walk],[app,
[dn,the],[dn,woman]]]]],[app,[dn,the],[dn,man]]]]],bill]


5  [this,man,walks] => s?
[app,[dn,walk],[app,[dn,this],[dn,man]]]


6  [this,man,walk] => s?


7  [this,men,walks] => s?


8  [this,men,walk] => s?
```

```
9  [these,man,walks] => s?


10  [these,man,walk] => s?


11  [these,men,walks] => s?


12  [these,men,walk] => s?
[app,[dn,walk],[app,[dn,these],[dn,men]]]


13  [the,man,that,walks] => np(_477)?
[app,[dn,the],[lmd,C,[both,[app,[dn,walk],C],[app,[dn,man],C]]]]


14  [the,man,that,walk] => np(_477)?


15  [the,men,that,walks] => np(_477)?


16  [the,men,that,walk] => np(_477)?
[app,[dn,the],[lmd,C,[both,[app,[dn,walk],C],[app,[dn,men],C]]]]


17  [the,man,that,suzy,likes] => np(_479)?
[app,[dn,the],[lmd,C,[both,[app,[app,[dn,like],C],[dn,suzy]],[app,[dn,man],C]]]]


18  [the,man,that,mary,thinks,walks] => np(_481)?
[app,[dn,the],[lmd,C,[both,[app,[app,[dn,think],[up,[app,[dn,walk],C]]],mary],
    [app,[dn,man],C]]]]


19  [the,man,that,mary,thinks,walk] => np(_481)?


20  [the,men,that,mary,thinks,walks] => np(_481)?


21  [the,men,that,mary,thinks,walk] => np(_481)?
[app,[dn,the],[lmd,C,[both,[app,[app,[dn,think],[up,[app,[dn,walk],C]]],mary],
    [app,[dn,men],C]]]]
```

```
22  [the,man,that,mary,thinks,bill,claims,suzy,likes] => np(_487)?
[app,[dn,the],[lmd,C,[both,[app,[app,[dn,think],[up,[app,[app,[dn,claim],
    [up,[app,[app,[dn,like],C],[dn,suzy]]]],bill]]],mary],[app,[dn,man],C]]]]


23  [the,man,that,mary,likes,the,woman,that,loves] => np(_487)?


24  [a,man,walks] => s?
[xst,C,[both,[app,[dn,man],C],[app,[dn,walk],C]]]


25  [john,likes,a,man] => s?
[xst,C,[both,[app,[dn,man],C],[app,[app,[dn,like],C],john]]]


26  [every,man,loves,a,woman] => s?
[all,C,[imply,[app,[dn,man],C],[xst,G,[both,[app,[dn,woman],G],
    [app,[app,[dn,loves],G],C]]]]]
[xst,C,[both,[app,[dn,woman],C],[all,G,[imply,[app,[dn,man],G],
    [app,[app,[dn,loves],C],G]]]]]


27  [every,man,claims,a,woman,sings] => s?
[all,C,[imply,[app,[dn,man],C],[app,[app,[dn,claim],[up,[xst,G,[both,
    [app,[dn,woman],G],[app,[dn,sing],G]]]]],C]]]
[all,C,[imply,[app,[dn,man],C],[xst,G,[both,[app,[dn,woman],G],
    [app,[app,[dn,claim],[up,[app,[dn,sing],G]]],C]]]]]
[xst,C,[both,[app,[dn,woman],C],[all,G,[imply,[app,[dn,man],G],
    [app,[app,[dn,claim],[up,[app,[dn,sing],C]]],G]]]]]


28  [john,claims,every,boy,thinks,a,woman,walks] => s?
[all,C,[imply,[app,[dn,boy],C],[app,[app,[dn,claim],[up,[app,[app,[dn,think],
    [up,[xst,G,[both,[app,[dn,woman],G],[app,[dn,walk],G]]]]],C]]],john]]]
[all,C,[imply,[app,[dn,boy],C],[app,[app,[dn,claim],[up,[xst,G,[both,[app,
    [dn,woman],G],[app,[app,[dn,think],[up,[app,[dn,walk],G]]],C]]]]],john]]]
[all,C,[imply,[app,[dn,boy],C],[xst,G,[both,[app,[dn,woman],G],[app,[app,
    [dn,claim],[up,[app,[app,[dn,think],[up,[app,[dn,walk],G]]],C]]],john]]]]]
[app,[app,[dn,claim],[up,[all,C,[imply,[app,[dn,boy],C],[app,[app,[dn,think],
    [up,[xst,G,[both,[app,[dn,woman],G],[app,[dn,walk],G]]]]],C]]]]],john]
[app,[app,[dn,claim],[up,[all,C,[imply,[app,[dn,boy],C],[xst,G,[both,[app,
    [dn,woman],G],[app,[app,[dn,think],[up,[app,[dn,walk],G]]],C]]]]]]],john]
[app,[app,[dn,claim],[up,[xst,C,[both,[app,[dn,woman],C],[all,G,[imply,[app,
    [dn,boy],G],[app,[app,[dn,think],[up,[app,[dn,walk],C]]],G]]]]]]],john]
[xst,C,[both,[app,[dn,woman],C],[all,G,[imply,[app,[dn,boy],G],[app,[app,
    [dn,claim],[up,[app,[app,[dn,think],[up,[app,[dn,walk],C]]],G]]],john]]]]]
```

```
[xst,C,[both,[app,[dn,woman],C],[app,[app,[dn,claim],[up,[all,G,[imply,[app,
    [dn,boy],G],[app,[app,[dn,think],[up,[app,[dn,walk],C]]],G]]]]],john]]]


29  [a,man,that,walks,sings] => s?
[xst,C,[both,[both,[app,[dn,walk],C],[app,[dn,man],C]],[app,[dn,sing],C]]]


30  [a,man,that,sings,walks] => s?
[xst,C,[both,[both,[app,[dn,sing],C],[app,[dn,man],C]],[app,[dn,walk],C]]]


31  [a,man,that,likes,every,woman,sings] => s?
[xst,C,[both,[both,[all,J,[imply,[app,[dn,woman],J],[app,[app,
    [dn,like],J],C]]]],[app,[dn,man],C]],[app,[dn,sing],C]]]


32  [john,likes,himself] => s?
[app,[app,[dn,like],john],john]


33  [john,likes,herself] => s?


34  [mary,likes,himself] => s?


35  [mary,likes,herself] => s?
[app,[app,[dn,like],mary],mary]


36  [john,thinks,mary,likes,himself] => s?


37  [the,boys,like,themselves] => s?
[app,[app,[dn,like],[app,[dn,the],[dn,boys]]],[app,[dn,the],[dn,boys]]]


38  [the,boy,likes,themselves] => s?


39  [the,boy,like,themselves] => s?


40  [the,boys,think,mary,likes,themselves] => s?
```

```
41  [john,walks,and,mary,sings] => s?
[both,[app,[dn,walk],john],[app,[dn,sing],mary]]


42  [john,walks,or,sings] => s?
[either,[app,[dn,walk],john],[app,[dn,sing],john]]


43  [john,or,mary,walks] => s?
[either,[app,[dn,walk],john],[app,[dn,walk],mary]]


44  [john,and,mary,walks] => s?


45  [john,and,mary,walk] => s?
[both,[app,[dn,walk],john],[app,[dn,walk],mary]]


46  [mary,thinks,john,walks,and,sings] => s?
[app,[app,[dn,think],[up,[both,[app,[dn,walk],john],[app,[dn,sing],john]]]],mary]
[both,[app,[app,[dn,think],[up,[app,[dn,walk],john]]],mary],[app,[dn,sing],mary]]


47  [mary,thinks,john,walks,or,sings] => s?
[app,[app,[dn,think],[up,[either,[app,[dn,walk],john],[app,[dn,sing],john]]]],mary]
[either,[app,[app,[dn,think],[up,[app,[dn,walk],john]]],mary],
    [app,[app,[dn,think],[up,[app,[dn,sing],john]]],mary]]
[either,[app,[app,[dn,think],[up,[app,[dn,walk],john]]],mary],[app,[dn,sing],mary]]


48  [bill,thinks,john,and,mary,walk] => s?
[app,[app,[dn,think],[up,[both,[app,[dn,walk],john],[app,[dn,walk],mary]]]],bill]


49  [bill,thinks,john,or,mary,walks] => s?
[app,[app,[dn,think],[up,[either,[app,[dn,walk],john],[app,[dn,walk],mary]]]],bill]
[either,[app,[app,[dn,think],[up,[app,[dn,walk],john]]],bill],
    [app,[app,[dn,think],[up,[app,[dn,walk],mary]]],bill]]


50  [the,man,that,walks,and,runs,sings] => s?
[app,[dn,sing],[app,[dn,the],[lmd,C,[both,[both,[app,[dn,walk],C],
    [app,[dn,run],C]],[app,[dn,man],C]]]]]


51  [the,man,that,walks,or,runs,sings] => s?
```

```
[app,[dn,sing],[app,[dn,the],[lmd,C,[both,[either,[app,[dn,walk],C],
    [app,[dn,run],C]],[app,[dn,man],C]]]]]


52  [the,man,that,john,sings,and,mary,likes] => np(sg(m))?


53  [the,man,that,john,likes,and,mary,sings] => np(sg(m))?


54  [the,man,that,john,likes,and,mary,loves] => np(sg(m))?
[app,[dn,the],[lmd,C,[both,[both,[app,[app,[dn,like],C],john],
    [app,[app,[dn,loves],C],mary]],[app,[dn,man],C]]]]


55  [the,man,that,john,sings,or,mary;likes] => np(sg(m))?


56  [the,man,that,john,likes,or,mary,sings] => np(sg(m))?


57  [the,man,that,john,likes,or,mary,loves] => np(sg(m))?
[app,[dn,the],[lmd,C,[both,[either,[app,[app,[dn,like],C],john],
    [app,[app,[dn,loves],C],mary]],[app,[dn,man],C]]]]


58  [a,woman,or,john,walks] => s?
[either,[xst,G,[both,[app,[dn,woman],G],[app,[dn,walk],G]]],[app,[dn,walk],john]]


59  [a,woman,walks,and,sings] => s?
[xst,C,[both,[app,[dn,woman],C],[both,[app,[dn,walk],C],[app,[dn,sing],C]]]]


60  [a,woman,walks,or,sings] => s?
[either,[xst,I,[both,[app,[dn,woman],I],[app,[dn,walk],I]]],
    [xst,I,[both,[app,[dn,woman],I],[app,[dn,sing],I]]]]
[xst,C,[both,[app,[dn,woman],C],[either,[app,[dn,walk],C],[app,[dn,sing],C]]]]


61  [every,woman,walks,and,sings] => s?
[all,C,[imply,[app,[dn,woman],C],[both,[app,[dn,walk],C],[app,[dn,sing],C]]]]


62  [every,woman,walks,or,sings] => s?
[all,C,[imply,[app,[dn,woman],C],[either,[app,[dn,walk],C],[app,[dn,sing],C]]]]
[either,[all,I,[imply,[app,[dn,woman],I],[app,[dn,walk],I]]],
    [all,I,[imply,[app,[dn,woman],I],[app,[dn,sing],I]]]]


 63  [john,claims,mary,walks,or,sings] => s?
[app,[app,[dn,claim],[up,[either,[app,[dn,walk],mary],[app,[dn,sing],mary]]]],john]
[either,[app,[app,[dn,claim],[up,[app,[dn,walk],mary]]],john],
    [app,[app,[dn,claim],[up,[app,[dn,sing],mary]]],john]]
[either,[app,[app,[dn,claim],[up,[app,[dn,walk],mary]]],john],[app,[dn,sing],john]]


64  [john,claims,every,woman,walks,or,sings] => s?
[all,C,[imply,[app,[dn,woman],C],[app,[app,[dn,claim],[up,[either,[app,
    [dn,walk],C],[app,[dn,sing],C]]]],john]]]
[all,C,[imply,[app,[dn,woman],C],[either,[app,[app,[dn,claim],[up,[app,
    [dn,walk],C]]],john],[app,[app,[dn,claim],[up,[app,[dn,sing],C]]],john]]]]
[app,[app,[dn,claim],[up,[all,C,[imply,[app,[dn,woman],C],[either,[app,
    [dn,walk],C],[app,[dn,sing],C]]]]]],john]
[app,[app,[dn,claim],[up,[either,[all,I,[imply,[app,[dn,woman],I],[app,
    [dn,walk],I]]],[all,I,[imply,[app,[dn,woman],I],[app,[dn,sing],I]]]]]]],john]
[either,[all,G,[imply,[app,[dn,woman],G],[app,[app,[dn,claim],[up,[app,
    [dn,walk],G]]],john]]]],[app,[dn,sing],john]]
[either,[all,I,[imply,[app,[dn,woman],I],[app,[app,[dn,claim],[up,[app,
    [dn,walk],I]]],john]]]],[all,I,[imply,[app,[dn,woman],I],[app,[app,
    [dn,claim],[up,[app,[dn,sing],I]]],john]]]]
[either,[app,[app,[dn,claim],[up,[all,G,[imply,[app,[dn,woman],G],[app,
    [dn,walk],G]]]]]],john],[app,[dn,sing],john]]
[either,[app,[app,[dn,claim],[up,[all,I,[imply,[app,[dn,woman],I],[app,
    [dn,walk],I]]]]]],john],[app,[app,[dn,claim],[up,[all,I,[imply,[app,
    [dn,woman],I],[app,[dn,sing],I]]]]]],john]]


65  [john,thinks,a,man,or,a,woman,sings] => s?
[app,[app,[dn,think],[up,[either,[xst,G,[both,[app,[dn,man],G],[app,[dn,sing],G]]],
    [xst,L,[both,[app,[dn,woman],L],[app,[dn,sing],L]]]]]]],john]
[either,[app,[app,[dn,think],[up,[xst,G,[both,[app,[dn,man],G],[app,[dn,sing],G]]]]],
   john],[app,[app,[dn,think],[up,[xst,L,[both,[app,[dn,woman],L],[app,[dn,sing],L]]]]],jc


66  [mary,or,suzy,likes,herself] => s?
[either,[app,[app,[dn,like],mary],mary],[app,[app,[dn,like],[dn,suzy]],[dn,suzy]]]


67  [mary,or,suzy,likes,themselves] => s?


68  [mary,or,suzy,like,herself] => s?
```

```
69  [mary,or,suzy,like,themselves] => s?


70  [john,and,bill,likes,himself] => s?


71  [john,and,bill,likes,themselves] => s?


72  [john,and,bill,like,himself] => s?


73  [john,and,mary,like,themselves] => s?
[both,[app,[app,[dn,like],john],john],[app,[app,[dn,like],mary],mary]]

no
| ?- ^Z
ruulo2% ^Z
script done on Thu Dec 20 16:05:12 1990
```

6 Comments on Coordination, by Guy Barry

A claim often made in favour of flexible categorial grammars is their ability to deal with a large proportion of coordination phenomena by means of a single principle, usually assumed to be as follows:

(104) Two (or more) strings may be coordinated to give a string of type X iff each has type X.

For the purposes of this discussion we shall treat the two directions of implication in (104) as two separate hypotheses, (105) and (106):

(105) If two (or more) strings can be given the same type X, then they can be coordinated to give a string of type X.

(106) If two (or more) strings can be coordinated to give a string of type X, then each can be given type X.

We shall assume (106) to be correct, so that for instance we posit some shared type for the conjuncts in (107):

(107) John is [lucky] and [a rogue].

However, in the framework of the Lambek calculus, (105) appears to be too general; as we shall see in the examples below, there are many cases where it is possible to give a

$A/C \Rightarrow A^{\wedge B}/C,\ A/B, B \Rightarrow A,\ B, A\backslash B \Rightarrow A, \forall Z.A \Rightarrow A[B/Z]$

Then this is a categorial derivation of the *de-re* interpretation of *a man* in *believes a man came in*:

$$\dfrac{\dfrac{\text{believes}}{\mathrm{VP/s}} \qquad \dfrac{\dfrac{\dfrac{\text{a man}}{\forall \mathrm{X.X/(X\backslash np)}}}{(\mathrm{s}^{\vee\mathrm{VP}})/((\mathrm{s}^{\vee\mathrm{VP}})\backslash \mathrm{np})}\forall\mathrm{E} \quad \dfrac{\dfrac{\text{came in}}{\mathrm{VP}}}{(\mathrm{s}^{\vee\mathrm{VP}})\backslash \mathrm{np}}\mathrm{vr}}{\mathrm{s}^{\vee\mathrm{VP}}}\mathrm{fa}}{\mathrm{VP}}\mathrm{ba}$$

The use of *vr* on the embedded verb turns around the usual function argument structure (using C as the translation of *came in*):

$\lambda z \lambda x_1^{tet} \lambda x_2^{e}[x_1(C(z))(x_2)]$
$\triangleright \lambda z \lambda X^{[tet,e]}[X_H(C(z))(X_{TH})]$

Hence the result for the embedded sentence is a function over the embedder:

$\lambda X^{[tet,e]}[GQ(\lambda z[X_H(C(z))(X_{TH})])]$
$\triangleright \lambda x_1^{tet} \lambda x_2^{e}[GQ(\lambda z[x_1(C(z))(x_2)])]$

Applied to the sentence embedder (using B as the translation of *believes*):

$\lambda x_2^{e}[GQ(\lambda z[\lambda p \lambda y[B(p)(y)](C(z))(x_2)])]$
$\triangleright \lambda x_2^{e}[GQ(\lambda z[B(C(z))(x_2)])]$

The mechanism that allowed this derivation was the value-raising sequent. It is derivable in the Lambek calculus, which is a calculus compactly expressing all the truths that follow from the intuitive understanding of the categorial connectives. There is therefore no way to change the system to make this value-raising sequent underivable without actually changing the meaning of the connectives. Thus one can claim an *emergent account*, for the polymorphic categorisation is necessitated to obtain the *basic* facts and it turns out that this is a sufficiently powerful categorisation that scope-ambiguity simply follows from the meaning of the connectives.

8 References

Barry and Pickering: 1990, 'Dependency and Constituency in Categorial Grammar', in Barry, Guy and Morrill, Glyn (eds.) *Studies in Categorial Grammar*, Edinburgh Working Papers in Cognitive Science, volume 5.

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.

van Benthem, Johan: 1986, 'Categorial Grammar', in *Essays in Logical Semantics*, Volume 8, Studies in Linguistics and Philosophy, D. Reidel, Dordrecht, pp. 123–150.

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

Buszkowski, W.: 1986, 'Generative Capacity of Non-Associative Lambek Calculus', Polish Academy (approx. reference).

Dosen, K.: 1989, 'Modal Logic as Metalogic', Research paper (approx. reference).

Emms, Martin: 1990, 'Polymorphic Quantifiers', in Barry, Guy and Morrill, Glyn (eds.) *Studies in Categorial Grammar*, Edinburgh Working Papers in Cognitive Science, volumeAmsterdam Colloquium.

Gazdar, G.: 1981, 'Unbounded dependencies and coordinate structure', *Linguistic Inquiry*, 12, 155-184.

Gazdar, G., Klein, E., Pullum, G. and Sag, I.: 1985, 'Generalized Phrase Structure Grammar', London: Basil Blackwell.

Girard, Jean-Yves: 1987, 'Linear Logic', *Theoretical Computer Science* **50**, 1–102.

Girard, Lafont, and Taylor: 1989, 'Proofs and Types', Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, Cambridge.

Hendriks, Herman: 1990, 'Flexible Montague Grammar', in Deliverable R1.2.A of DYANA Dynamic Interpretation of Natural Language, ESPRIT Basic Research Action BR3175.

Hepple, Mark: 1990a, 'Normal form theorem proving for the Lambek calculus', in Karlgren, H. (ed.), Proceedings of COLING 1990.

Hepple, Mark: 1990b, 'Word Order, Binding and Extraction in Categorial Grammar', PhD thesis, Centre for Cognitive Science, University of Edinburgh. (approx. reference).

Hepple, Mark and Glyn Morrill: 1989, 'Parsing and Derivational Equivalence', *Proceedings of the Fourth Conference of the European Chapter of the Association for Computational Linguistics.*

König, E.: 1989, 'Parsing as natural deduction', *Proceedings of the Annual Meeting of the Association for Computational Linguistics.*

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

Lambek, J.: 1961, 'On the calculus of syntactic types', in R. Jakobson (ed.), Studies of Language and its Mathematical Aspects, American Mathematical Society, pp. 166–178.

Lewin, I.: 1990, 'An algorithm for quantifier scoping', Technical Report, Dept of Artificial Intelligence, University of Edinburgh.

Moortgat, Michael: 1990, 'The Quantification Calculus: Questions of Axiomatisation', in Deliverable R1.2.A of DYANA Dynamic Interpretation of Natural Language, ESPRIT Basic Research Action BR3175.

Morrill, Glyn: 1988, 'Extraction and Coordination in Phrase Structure Grammar and Categorial Grammar', PhD Thesis, Centre for Cognitive Science, University of Edinburgh.

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

Morrill, Glyn: 1990a, 'Grammar and Logical Types', in Barry, Guy and Morrill, Glyn (eds.) *Studies in Categorial Grammar*, Edinburgh Working Papers in Cognitive Science, volumeAmsterdam Colloquium.

Morrill, Glyn: 1990b, 'Intensionality and Boundedness', to appear in *Linguistics and Philosophy*.

Morrill, Glyn, Leslie, Neil, Hepple, Mark and Barry, Guy: 1990, 'Categorial Deductions and Structural Operations', in Barry, Guy and Morrill, Glyn (eds.) *Studies in Categorial Grammar*, Edinburgh Working Papers in Cognitive Science, volume

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.

Oehrle and Zhang: 1989, ms. University of Tuscon. (approx. reference).

Partee, Barbara and Mats Rooth: 1983, 'Generalized conjunction and type ambiguity', in R. Bäuerle, C. Schwarze and A. von Stechow (eds.), *Meaning, Use, and Interpretation of Language*, Volume 6, Linguistic Analysis, de Gruyter, Berlin, pp. 53–95.

Pollard and Sag: 1989, 'An information-based approach . . . ', CSLI. (approx. reference).

Prijatelj, Andreja: 1989, 'Intensional Lambek Calculi: Theory and Application', Prepublication Series 89–06, Institute for Language, Logic and Information, University of Amsterdam.