Proceedings

of

Formal Grammar 1999

Geert-Jan M. Kruijff and Richard T. Oehrle
(editors)

Chapter 1

Towards a minimal logic for minimalist grammars:

a transformational use of Lambek calculus

A. Lecomte & C. Retoré

Abstract. Many convergence points have been observed during the recent years between the Minimalist Program and the program of Categorial Grammar, above all since the formalization of minimalist ideas by E. Stabler. For instance, the *Merge*-operation is exactly like functional application. Moreover the fundamental operation of *feature-checking*, which is at the basis of the *Move*-operation, can really be depicted as a resource consumption procedure, something familiar to so called *resource conscious logics*. This makes rise a deep interest in looking for a logical formulation of minimalist grammars. Such an enterprise is not done for the sake of spurious formalization. If we take the chomskyan framework seriously, it seems natural to assume that UG consists in a very general set of principles that must be expressed in the most condensed way, and that derivations are made of steps of a few different sorts exactly like it is the case in a logic.

1.0.1 The convergence of the minimalist program and categorial grammar

Both generative grammar and categorial grammar postulate the existence of a universal grammar. In generative grammar, this universal grammar is supposed to be provided by a set of *principles*, now reduced to a small number, among which are: the structure dependence principle, the *Merge* and *Move* operations, the Binding principles, the Head Movement Constraint... Particular languages are obtained by assigning values to so called *parameters*. Because of the evolution of the theory, and notably the fact that *moves* are assumed to be always triggered by the operation of *feature checking*, these parameters are simply boolean values assigned to the strength of features.
On the categorial grammar side, e.g. in Moortgat setting [8, 9], it is assumed that UG is provided by a *base logic*, which puts together several connective families $\bullet_i$, $/_i$, $\backslash_i$, connected by *communication postulates*, and that only this set of postulates may change according to the language to be learnt: the ground logic remains invariant, and is thus supposed to capture all the universal principles.
From our viewpoint, Chomsky's conception is very appealing because of the simple nature of the parameters which are postulated. The features by means of which communications are established between the sentence

Proceedings of Formal Grammar 1999
Geert-Jan M. Kruijff & Richard T. Oehrle (editors).

constituants may be seen as nodes in a net, and their relative strengths as connection weights, thus evoking some connectionnist aspects, but Universal Grammar remains unclear. On the contrary, the conception of UG in categorial grammar is very clear and appealing: it is in fact natural to assume that a so general and abstract device may be seen as a kind of logic, simply because logics study abstract symbolic systems, but language variation seen as a changing set of postulates is not entirely satisfactory: it seems hard to assume that a language is learnt by learning abstract postulates of this kind.

This motivates the attempts to conciliate the two approaches. What we aim to find is a very limited set of rules that would be sufficient in order to give an account of *Merge* and *Move*, and that would be sufficiently restricted in order to incorporate at least some of the principles (like *economy constraints*), in such a way that there would be no need for their independent formulation.

We assume in this paper that each item of the lexicon consists in a set of features, which are divided as follows:

- Phonetic features for example */speaks/,/linguist/,/some/*, ...
- Semantic features for example *(speaks),(linguist),(some)*, ...
- Syntactic or formal features:
  - categorial features (categories) involved in *merge*:
    $\textsc{base} = \{\mathtt{c}, \mathtt{t}, \mathtt{v}, \mathtt{d}, \mathtt{n}, \ldots\}$
  - functional features involved in *move*:
    $\textsc{fun} = \{\overline{\mathtt{k}}, \overline{\mathtt{K}}, \overline{\mathtt{wh}}, \ldots\}$

1.0.2 Stabler's minimalist grammars

In his paper *Derivational Minimalism* [10] and in the subsequent *Remnant movement and structural complexity* [11], Stabler provides formal grammars which realize the view of grammar expressed in the Minimalist Program. Lexical entries are ordered sequences in:

$\textsc{label} = \textsc{select}^* \ (\textsc{licensors}) \ \textsc{select}^* \ \textsc{base} \ \textsc{licensees} \ P^* \ I^*$

- P phonetic features
- I semantic features
- $\textsc{select} = \{=\mathtt{b}, =\mathtt{B}, \mathtt{B}{=} | \mathtt{b} \in \textsc{base}\}$ select a category
- $\textsc{licensees} = \{-\mathtt{x} | \mathtt{x} \in \textsc{fun}\}$ needs a move feature
- $\textsc{licensors} = \{+\mathtt{x}, +\mathtt{X} | \mathtt{x} \in \textsc{fun}\}$ provides a move feature

The structures he is using are binary trees, with internal nodes labelled $<$ or $>$ which indicates where the head of the (sub)tree is to be found. These trees can be interpreted as 'T-markers', that term being adapted from Chomsky (1955/75) — structures which record the history of a derivation, including the transformational phase (see also Cornell [4]).

In Stabler grammars, there are two kinds of merge, and one kind of movement, and they are completely determined by the sequence of features at the head of the T-marker. Merge is defined between two T-markers u and t the head of u starting with $=\mathtt{x}$ and the head t starting with x with $x \in$ BASE; let u' (resp. t') denote u (resp. t) in which the $=\mathtt{x}$ (resp. x) feature starting the head is cancelled.

- if u is a lexical item then the resulting tree is $u' < t'$ (so u' is the head and is on the left)
- otherwise the resulting tree is $t' > u'$ (so u' is the head in this case as well, but it is on the right)

Roughly speaking, movement is defined as follows: assume that at the leftmost position (spec* position) we have a $+\mathtt{x}$ and that at the rightmost (comp$^+$ position) we have a $-\mathtt{x}$: then the movement takes the whole constituent having $-\mathtt{x}$ as a head and moves it to the leftmost position (spec* position).

1.0.3 Why do we want a logical formalization?

Before stating the advantages that would bring a logical formalization, let us observe its naturality. There is a striking similarity between categorial grammar and minimalist grammars:

- both *merge* and *move*, in the minimalist syntax, are governed by resource consumption, which suggest a formulation within resource sensitive logics
- there is a clear parallel between *function* and *head* (or between *argument* and *non-head*).

The first advantage of a logical formalization is a simplification which can be stated as a *radical lexicalization*: both merge and move make reference to the tree structure and in particular to the head of the T-marker while a logical formulation of rules should only depend on the roots of the trees: all structure under the root can be erased.

Logic is also attractive because trees that we obtain in logic, which represent proofs in a Natural Deduction system, can be easily translated into logical forms, an objective that is pursued as well in generative grammar as in categorial grammar. The Curry-Howard homomorphism has often be used in this perspective. Indeed, in our opinion, "pure" resource logic alone cannot describe language: language-specific notions and the logic-based computing device have to be integrated as "harmoniously" as possible.

1.0.4 Minimalist grammars within the Lambek calculus

We must here warn the reader that *although we use Lambek calculus [7], we do* not *use Lambek grammars* (Lambek grammars are defined by $m_1 \ldots m_n \in L$ iff for each i there exists a type T_i of m_i such that $T_1 \ldots T_n \vdash S$).
In our presentation, we will derive the start symbol of the grammar (here $\mathtt{c}$) from the types of the words, but we depart from the ordinary use of Lambek grammars in viewing types as *closed deductions* rather than hypotheses [1]. There are therefore no hypotheses, nor any order on hypotheses which depict word order. Actually, word order is computed by means of *labels* propagation plus a small device (like an automaton) which can erase phonological content after copying. Let us start, for an example, from the following translation of Stabler types into Lambek formulas:

entry	*label*	*type*
every	$=\mathtt{n}\ \mathtt{d}\ -\overline{\mathtt{k}}$ *every*	$((\mathtt{d} \otimes \overline{\mathtt{k}})/\mathtt{n})$
some	$=\mathtt{n}\ \mathtt{d}\ -\overline{\mathtt{k}}$ *some*	$((\mathtt{d} \otimes \overline{\mathtt{k}})/\mathtt{n})$
language	$\mathtt{n}$ *language*	n
linguist	$\mathtt{n}$ *linguist*	n
speaks	$=\mathtt{d}\ +\overline{\mathtt{k}}\ =\mathtt{d}\ \mathtt{v}$ *speaks*	$((\overline{\mathtt{k}}\backslash(\mathtt{d}\backslash\mathtt{v}))/\mathtt{d})$
(*tense*)	$=\mathtt{v}\ +\overline{\mathtt{k}}\ \mathtt{t}$	$((\overline{\mathtt{k}}\backslash\mathtt{t})/\mathtt{v})$
(*comp*)	$=\mathtt{t}\ \mathtt{c}$	$(\mathtt{t}\backslash\mathtt{c})$

where strings (like *language*, *speaks* and so on) represent labels for deductions of the corresponding Lambek formulae.
For instance, to say that $((\mathtt{d} \otimes \overline{\mathtt{k}})/\mathtt{n})$ is associated with *some* is to say that we assume the closed deduction:

$$\vdash \textit{some} : ((\overline{\mathtt{k}} \otimes \mathtt{d})/\mathtt{n})$$

We use two steps in the computation:

1. logic (Lambek calculus) expresses constituents structure and head/non-head relation ship
2. a simple automaton reconstructs strings out of the proof in the Lambek calculus.

[1]A *closed deduction* is a deduction which uses no hypotheses. In Lambek grammars, we say that types of words are *hypotheses* because they are displayed on the left-hand side of the deduction relation expressed by one sequent. In $T_1 \ldots T_n \vdash S$, $T_1 \ldots T_n$ are hypotheses, from which S is deduced. In our system words (= constants) are not associated with hypotheses, and therefore there is no order on hypotheses like it is the case in Lambek grammars.

The game is to build a proof of c, using closed deductions associated with lexical items. Out of the proof of c, in order to compute the word sequence, we propagate labels. When starting, only the lexical items are endowed with a phonological labelling. Then in "application rules" (or modus ponens, or residuation) the label of the node is simply the concatenation of the two labels, but in product elimination the label is duplicated or split into two parts according to the types. That's where, for instance the difference between strong case and weak case takes place. This mimicks movements.
It should be observed that to consider the lexical items as closed proofs rather than hypotheses allows us to consider as consecutive two hypotheses which are only separated by lexical items. We thus avoid that movements cross each other, in accordance with the Head Movement Constraint. Of course, that's probably too strong a constraint because we know that some movements can cross (a phrasal movement can cross a head movement). It is the reason why we shall necessarily have to extend our calculus towards some kind of *mixed* system.
For the time being, let us make these ideas more precise and let us state the product elimination rule in the Gentzen natural deduction style[2] :

$$\frac{\Delta \vdash \alpha : X \otimes Y \quad \Gamma, (x : X), (y : Y), \Gamma' \vdash t : C}{\Gamma, \Delta, \Gamma' \vdash [\alpha/x, \alpha/y]t : C}\,[\otimes E]$$

In our applications of this rule, Δ will in fact be *empty*, Γ or Γ' can also be considered empty. Hypotheses are always labelled with *variables*.
Within the tree-like natural deduction *à la* Prawitz, this rule is stated as follows. Let U and V be hypotheses (expressed by variables) and ℓ and ℓ' be deduced without any free hypothesis. When the rule is applied, both U and V are cancelled, thus acquiring the ^{i}-index.

$$\frac{\begin{array}{c}\ell\\ \vdots\\ U \otimes V\end{array} \qquad \begin{array}{c}\ell_l\ U^i\ \ell'\ V^i\ \ell_r\\ \vdots\\ C\end{array}}{C}\,[\otimes E]^i$$

All the phonological and semantical forms appearing in the proof (but the lexicon items) are computed from the structure of the proof and the lexical items. The fact that the labelling terminates is ensured by the fact that there is no cycle when adding arcs from an eliminated product formula to the discharged hypotheses in a wellformed natural deduction. As said above, following Stabler, we denote the semantics of some expression within braces, and its phonological part within slashes. When none of these parentheses occurs, this simply means that both the semantics and the phonology lie

[2] In fact, the precise substitutions depend on the characteristic *weak* or *strong* of the feature, here X. If X is weak, only the semantical part of α is substituted to x: this will precisely be the work done by the extra-logical device

at the same place. When reading the derived string, the second time we meet a complete item, this should be considered a trace, in conformity with the copy theory[3]. This allows a representational description as opposed to a derivational one: that's indeed what we are using here since logic prefers static representations. From the linguist's point of view, this amounts to prefer an operation like *Form Chain* to *Move*. The strings we obtain are like:

```
every linguist every linguist (some language) speaks some
language
```

Such a sentence in a Stabler-like representation (which uses movement rather than copy) would be:

```
(every linguist) /every linguist/ (some language) speaks
/some language/
```

which yields the logical form (correct quantifier raising)[4]:

```
(every linguist) (some language) (speaks ...)
```

and the phonological form:

```
every linguist speaks some language.
```

according to the following deduction that, for reasons of size we cut off into two pieces. The first piece gives a reduction of `speaks some language` to `v`, and the second piece shows the continuation of the proof, by using the conclusion thus obtained.

$$\cfrac{\begin{matrix} y^2 \\ \mathtt{d}^2 \end{matrix} \qquad \cfrac{\cfrac{\begin{matrix}\text{some}\\ ((\overline{\mathtt{k}}\otimes \mathtt{d})/\mathtt{n})\end{matrix} \quad \begin{matrix}\text{language}\\ \mathtt{n}\end{matrix}}{\begin{matrix}\textit{some language}\\ \overline{\mathtt{k}}\otimes \mathtt{d}\end{matrix}}[/E] \qquad \cfrac{\begin{matrix} x^1 \\ \overline{\mathtt{k}}^1\end{matrix} \quad \cfrac{\begin{matrix}\text{speaks}\\ ((\overline{\mathtt{k}}\backslash(\mathtt{d}\backslash\mathtt{v}))/\mathtt{d})\end{matrix} \quad \begin{matrix} y^1\\ \mathtt{d}^1\end{matrix}}{\begin{matrix}\textit{speaks } y^1\\ (\overline{\mathtt{k}}\backslash(\mathtt{d}\backslash\mathtt{v}))\end{matrix}}[/E]}{\begin{matrix} x^1\ \textit{speaks } y^1\\ (\mathtt{d}\backslash\mathtt{v})\end{matrix}}}{\begin{matrix}\textit{(some language)}\\ \textit{speaks some language}\\ (\mathtt{d}\backslash\mathtt{v})\end{matrix}}[\otimes E]^1}{\begin{matrix} y^2\ \textit{(some language)}\\ \textit{speaks some language}\\ \mathtt{v}\end{matrix}}[\backslash E]$$

[3]in case there are several items with the same phonological form, one should of course distinguish them by indices, which is possible in our procedure but not needed in our little examples. (cf figure)

[4]to get the arguments of *speaks* correct, the verb should have been modelled with more details, using a VP-shell and moving the subject out of it; this is not at all a problem but makes the example even bigger.

And the continuation of the proof is :

$$
\dfrac{
  \dfrac{\begin{matrix}\text{every}\\ ((\overline{\mathtt{k}}\otimes\mathtt{d})/\mathtt{n})\end{matrix} \quad \begin{matrix}\text{linguist}\\ \mathtt{n}\end{matrix}}{\begin{matrix}\textit{every linguist}\\ \overline{\mathtt{k}}\otimes\mathtt{d}\end{matrix}}[/E]
  \quad
  \dfrac{\begin{matrix}x^2\\ \overline{\mathtt{K}}^2\end{matrix} \quad \dfrac{\begin{matrix}\emptyset\text{: tense}\\ ((\overline{\mathtt{K}}\backslash\mathtt{t})/\mathtt{v})\end{matrix} \quad \begin{matrix}y^2\ \textit{(some language)}\\ \textit{speaks some language}\\ \mathtt{v}\end{matrix}}{\begin{matrix}y^2\ \textit{(some language)}\\ \textit{speaks some language}\\ (\overline{\mathtt{K}}\backslash\mathtt{t})\end{matrix}}[/E]}{\begin{matrix}x^2\ y^2\ \textit{(some language) speaks some language}\\ \mathtt{t}\end{matrix}}[\backslash E]
}{
  \begin{matrix}\textit{every linguist every linguist}\\ \textit{(some language) speaks some language}\\ \mathtt{t}\end{matrix}
}[\otimes E]^2
$$

Out of this example one can use Chomskyan explanation to cross linguistic variations: assume that the verb is in fact a strong case assigner, then not only the semantics of *some language* would move but also the phonological features. We thus would obtain:

```
every linguist every linguist some language speaks some
language.
```

which yields the semantical reading[5]:

```
(every linguist) (some language) (speaks ...)
```

and has the phonological form:

```
every linguist some language speaks.
```

1.0.5 A necessary extension

VSO languages are much more difficult to obtain, and in fact they cannot be obtained in this fragment of the Lambek calculus. The reason is that necessarily such a language involves a *head-movement* which *crosses* the phrasal movements, something which is forbidden by the present state of the calculus.
A suggestion to solve this problem is to use the special unary connective that M. V. Abrusci [1] has introduced in *non commutative intuitionistic linear logic without exponential* (N-LLI$^-$), a system which is in fact a conservative extension of the Lambek calculus [6].

[5] cf. previous footnote.

[6] But we shall also explore in a future work other solutions using partially commutative linear logic, in the directions indicated by [5] and [2]

From now on, we assume the connective $\triangleright$ associated with all the licensees and only to them (not to the select features for instance), and we shall restrict ourselves to some particular proofs in the space of all proofs: those proofs which enjoy the property of what we call: *move-admissibility*, that is a property according to which every hypothesis must be discharged *as soon as possible* in the dynamic of the proof.
As a matter of example, let us see our translation of Stabler's types for VSO languages:

$entry$	$Stabler'stype$	$label : type$
$Peter$	$\mathtt{d}\ -\overline{\mathtt{k}}\ peter$	$peter : \overline{\mathtt{k}} \otimes \mathtt{d}$
$english$	$\mathtt{d}\ -\overline{\mathtt{k}}\ english$	$english : \overline{\mathtt{k}} \otimes \mathtt{d}$
$speaks$	$=\mathtt{d}\ +\overline{\mathtt{k}}\ =\mathtt{d}\ \mathtt{v}\ speaks$	$speaks : \mathtt{V} \otimes ((\overline{\mathtt{k}}\backslash(\mathtt{d}\backslash\mathtt{v}))/\mathtt{d})$
$(tense)$	$=\mathtt{V}\ +\overline{\mathtt{K}}\ \mathtt{t}$	$((\mathtt{T} \otimes ((\overline{\mathtt{K}}\backslash\mathtt{t})/\mathtt{v}))/\mathtt{V})$
$(comp)$	$=\mathtt{T}\ \mathtt{c}$	$((\mathtt{c}/\mathtt{t})/\mathtt{T})$

where capital letters denote strong select features, and functional features (like $\overline{\mathtt{k}}$) are assumed to be affected by the exchange modality.
And we show an example of proof (in two pieces) under the form of a tree similar to a T-marker. (cf. figure) In the result, repetitions are omitted, thus producing:

```
/speaks//peter/(peter)(english)(speaks)/english/
```

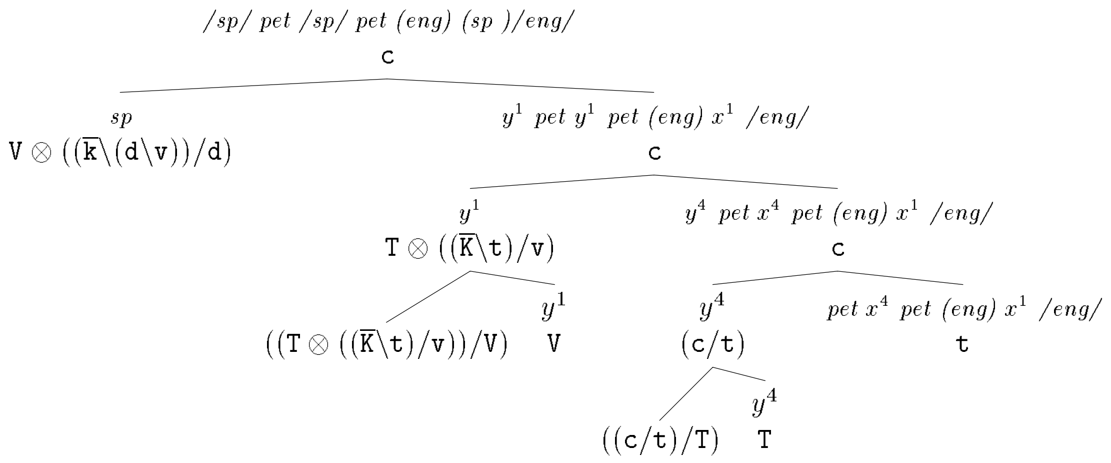
thus providing the following PF and LF:

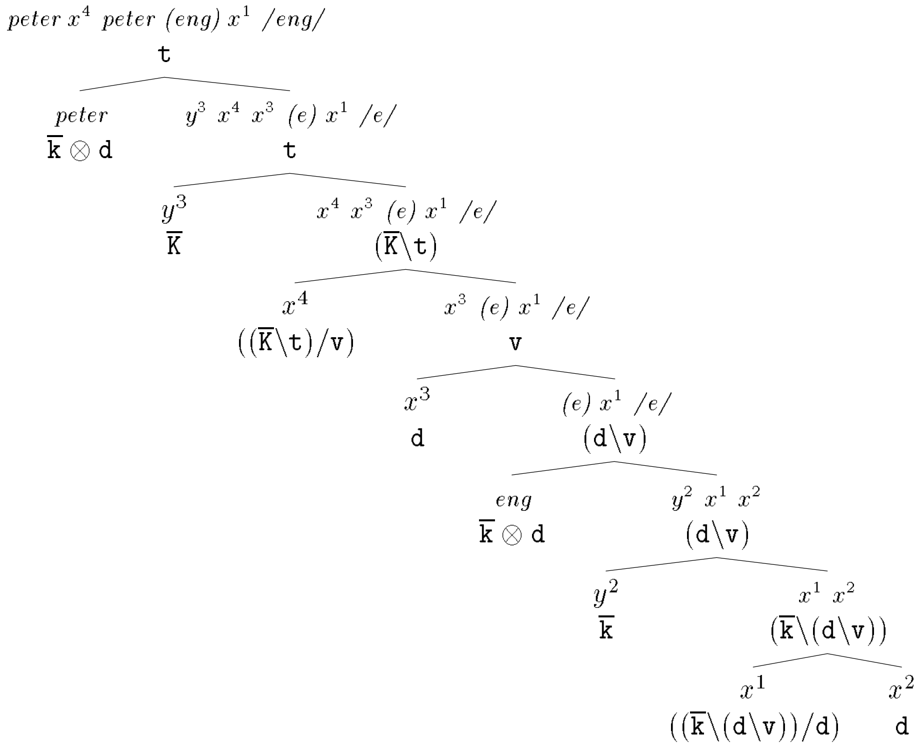
```
/speaks peter english/

(peter)(english)(speaks)
```

1.0.6 Conclusion and ongoing work

We hope the reader to be convinced by the simplicity of this system which is achieved by separating the hierarchical constituent structure and word order. Word order results from movement in the Chomskyan tradition, and from reading this structure with a simple automaton which takes into account more specific language properties. Thus we can work within the pure Lambek calculus, and obtain structures which are extremely close to the T-markers used in the generative tradition. Moreover, we shall demonstrate in future work that languages like $\mathtt{c}^n\mathtt{v}^n\mathtt{s}^n\mathtt{o}^n$ or $\mathtt{c}^n\mathtt{v}^n\mathtt{x}^n\mathtt{s}^n\mathtt{o}^n$ can be generated in this framework and more generally, that it is possible to prove an equivalence result between Stabler's grammars and our "New" Lambek grammars, thus getting the generative power needed for mildly context-sensitive languages.


/sp/ pet /sp/ pet (eng) (sp)/eng/
c
sp
V ⊗ ((k̄\(d\v))/d)
y¹ pet y¹ pet (eng) x¹ /eng/
c
y¹
T ⊗ ((K̄\t)/v)
y⁴ pet x⁴ pet (eng) x¹ /eng/
c
((T ⊗ ((K̄\t)/v))/V)
y¹
V
y⁴
(c/t)
pet x⁴ pet (eng) x¹ /eng/
t
((c/t)/T)
y⁴
T


peter x⁴ peter (eng) x¹ /eng/
t
peter
k̄ ⊗ d
y³ x⁴ x³ (e) x¹ /e/
t
y³
K̄
x⁴ x³ (e) x¹ /e/
(K̄\t)
x⁴
((K̄\t)/v)
x³ (e) x¹ /e/
v
x³
d
(e) x¹ /e/
(d\v)
eng
k̄ ⊗ d
y² x¹ x²
(d\v)
y²
k̄
x¹ x²
(k̄\(d\v))
x¹
((k̄\(d\v))/d)
x²
d

Bibliography

[1] M. Abrusci. Exchange connectives for non commutative intuitionistic linear logic. in Abrusci, Casadio, Moortgat (eds). *Linear Logic and Lambek Calculus*, DYANA Occ. Pub., Rome-Utrecht, 1993.

[2] M. Abrusci & P. Ruet. Commutativity and non-commutativity. 1998.

[3] Noam Chomsky. *The minimalist program.* MIT Press, Cambridge, MA, 1995.

[4] T. Cornell. Derivational and Representational views of minimalist transformational grammar. In A. Lecomte, F. Lamarche, G. Perrier (eds) *Logical Aspects of Computational Linguistics, LACL97*, LNCS-LNAI, 1582, Springer, 1999.

[5] P. de Groote. Partially commutative linear logic: sequent calculus and phase semantics. in V. Abrusci and C. Casadio (eds) *Proofs and Linguistics Categories, Proceedings of the 1996 Roma Workshop.* CLUEB, Bologna, 1996.

[6] Richard Kayne. *The Antisymmetry of Syntax.* Number 25 in Linguistic Inquiry Monographs. M.I.T. Press, Cambridge, Massachusetts, 1994.

[7] Jim Lambek. The mathematics of sentence structure. *American Mathematical Monthly*, 65. pages 154–170, 1958.

[8] Michael Moortgat. Categorial type logic. In J. van Benthem and A. ter Meulen, *Handbook of Logic and Language*, Elsevier, 1996. chapter 2, pages 93–177.

[9] M. Moortgat, 'Constants of Grammatical Reasoning', to appear, CSLI, 1999.

[10] Edward Stabler. Derivational minimalism. In C. Retoré, editor, *Logical Aspects of Computational Linguistics, LACL'96*, volume 1328 of *Lecture Notes in Artificial Intelligence.* Springer-Verlag, 1997.

[11] E. Stabler, 'Remnant Movement and Structural Complexity', to appear, CSLI, 1999.
http://www.humnet.ucla.edu/humnet/linguistics/people/stabler/stabler.htm