16

Remnant movement and complexity

EDWARD STABLER

16.1 Remnant movement

Linguists in the transformational tradition have recently been exploring the extent to which previous empirical coverage can be maintained with fewer grammatical devices. For example, Kayne (1994) develops some analyses in a framework that requires constituents to begin in specifier-head-complement order, with only leftward movement, and Chomsky (1995) explores how the structure building operations in such a system can be bottom-up and driven by lexical features. In more recent work, Kayne (1998) begins the exploration of the possibility that there is no covert movement, and Koopman and Szabolcsi (1998) show how head movement is not needed in the analysis of verbal complexes.

With the restricted grammatical options these theories make available, "remnant movement" has come to have increasing prominence, where a "remnant" is a constituent from which material has been extracted. Moving a constituent from which material has already been extracted means that traces of earlier movements may be carried to positions where they are no longer c-commanded by their antecedents, something which was banned in earlier theories. Interest in remnant movement has a longer history. For example, den Besten and Webelhuth (1990) proposed analyses like the following:

(1) $[t_i$ gelesen$]_j$ hat Hans [das Buch]$_i$ nicht t_j
read has Hans the book not

Similar analyses are considered in Webelhuth (1992) and Müller (1996).[1] Nkemnji (1995) uses remnant movement in his analysis of the Bantu language Nweh,

(2) njikem a ke? [te t_i akendɔŋ]$_j$ pfɛt$_i$ t_j
he Agr P1 neg plaintains eat

And Koopman (1996) sketches a range of related analyses:

(3) a. who$_i$ do you think [t_i came]$_j$ t_i t_j?
b. who$_i$ [t_i came]$_j$ do you think t_i t_j?

With the renewed interest in remnant movement coming from theoretical shifts, a wide range of constructions involving negation and related things is considered in Kayne (1998), where it is proposed that alternative analyses like the following yield scope ambiguities:

(4) a. I will force you to [marry t_i]$_j$ [no one]$_i$ t_j
b. I will [force you to marry t_i]$_j$ [no one]$_i$ t_j

The first analysis of this example yields a narrow object scope reading, with the negative DP raising to the specifier of NegP, followed by a predicate raising step (cf. Koster 1994, Zwart 1994, 1997). The second, wide object scope analysis has longer movements.

The best-worked out remnant movement analysis is the account of verbal complex formation in Koopman and Szabolcsi (1998). Developing an observation of Kenesei, Koopman and Szabolcsi observe the following pattern in negated or focused sentences of Hungarian, schematized on the right where "M" is used to represent the special category of "verbal modifiers" like *haza-*:

(5) Nem fogok akarni kezdeni haza-menni V1 V2 V3 M V4
not will-1s want-inf begin-inf home-go-inf

(6) Nem fogok akarni haza-menni kezdeni V1 V2 M V4 V3
not will-1s want-inf home-go-inf begin-inf

(7) Nem fogok haza-menni kezdeni akarni V1 M V4 V3 V2
not will-1s begin-inf want-inf home-go-inf

All of these sentences mean roughly the same thing: "I won't want to begin to go home." Notice that the verbs following the modifier appear in inverted order. Other orders are not possible (on the readings indicated by the numbering of the verbs). In particular, it is

[1] An alternative analysis is provided in Fanselow (1987), and a brief critical survey of remnant movement analyses is provided in De Kuthy and Meurers (1998).

interesting that while the pattern above suggests that a constituent [M V4 V3] can invert with V2 to yield [M V4 V3 V2], the uninverted form [V3 M V4] which we see in (5) cannot invert with V2 to yield V1 V3 M V4 V2.

Koopman and Szabolcsi (1998) propose a very natural and uniform analysis of this paradigm which can be schematically represented as follows:

(8)	V1 V2 V3 [V4 M] ⇒	(not a surface form)
	V1 V2 [V3 [M_1 [V4 t_1]]] ⇒	=(5)
	V1 [V2 [M_1 [V4 t_1]]$_2$ [V3 t_2]] ⇒	=(6)
	V1 [M_1 [V4 t_1]]$_2$ [V3 t_2]]$_3$ [V2 t_3]	=(7)

Notice that the various inverted orders are obtained by "rolling up" the predicates into increasingly complex constituents. Let's say that a grammar *rolls up* some category XP iff one XP_1 can move to a higher XP_2, with XP_2 in turn moving to a higher XP_3, and so on without bound. In the analysis (8), we see VP rolling up.

The verbal modifier M can be phrasal (e.g. *a szoba-ba* 'the room-into'), and so it is natural to assume that all the movements indicated here are phrasal. In fact, Koopman and Szabolcsi (1998) show that the idea that there are two different kinds of movement, head movement and phrasal movement, faces serious problems in the account of verbal complexes in Hungarian and Germanic: our best accounts do not work, and often we seem to need parallel accounts for very similar constructions that contain more than just heads.

These analyses have interesting formal properties, which can be revealed when they are cast into a natural generative framework.

16.2 Languages as closures

As in Keenan and Stabler (1996), we define a grammar as a pair $G = \langle Lex, \mathcal{F} \rangle$, where the lexicon Lex is a set of expressions and $\mathcal{F}$ is a set of structure building functions (functions from tuples of expressions to expressions). Then let the language defined by the grammar be the set of all expressions that can be built up from the lexicon by the structure building rules, $L(G) = closure(Lex, \mathcal{F})$.[2] In typical linguistic applications, as in this paper, both Lex and $\mathcal{F}$ are finite.

Given a grammar that is formalized in this way, we will consider:

[2]Keenan and Stabler (1996) define languages this way in order to consider, for example, what relations are preserved by automorphisms of L that fix $\mathcal{F}$, just as as in semantics we can consider the similar question: what is preserved by automorphisms of E that fix $E, (2, \leq)$. But in this paper, languages are defined as closures just for simplicity.

• What is the expressive power of grammars that derive the verbal complexes? (what sets are definable)

• What is the structural complexity of the verbal complexes in these grammars?

• Since structure building is driven by lexical features, is there a useful representation of derivations as graphs of feature checking relations?

Expressive power is familiar, but expression complexity is perhaps not so familiar, so we quickly review the ideas we will use. What is the size of a sentence like the following?

every student criticized some teacher

When comparing sentences, one possible measure is simply the count of the characters, the length of the sequence. In this case, we have 37 characters (counting the spaces between words). To get a slightly more universal measure, it is common to consider how many binary choices are required to specify each element of the sequence. In the ASCII coding scheme, each character is specified by 7 binary choices, 7 bits. So in the ASCII coding scheme, the sequence is represented with 359 bits.[3]

When we have a grammar that includes the sentence, we make available another way of specifying the sentence. The sentence can be specified by specifying its shortest derivation. Let's see how this works. Suppose we have the following grammar $G = \langle Lex, \mathcal{F} \rangle$, with 8 lexical items consisting of string-category pairs, and 4 structure building rules which operate as indicated here:

Lex:

every D	teacher N	praised V	and CONJ
some D	student N	criticized V	or CONJ

$\mathcal{F}$:

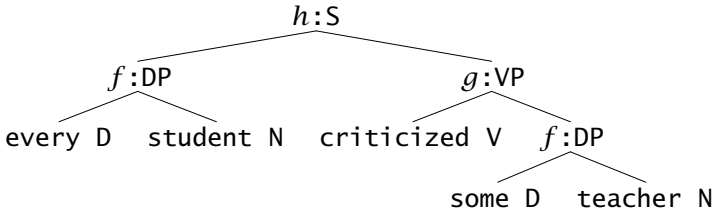
f: s D, t N $\longmapsto$ st DP h: s DP, t VP $\longmapsto$ st S

g: s V, t DP $\longmapsto$ st VP i: s S, t CONJ, u S $\longmapsto$ stu S

For example the function f takes a pair of expressions where the first is the string s of category D and the second is a string t of category N to yield the expression which is the concatenation st of category NP. The language $L(G)$ is clearly infinite, because we can form arbitrarily

[3] The ASCII coding scheme is of course not an "optimal code" for a source of English text. We just use it as an example here because it is a well-known, standard code.

large sentential conjunctions, and it includes a derivation that can be depicted with a tree like the following:



The written form of this sentence is specified by 359 bits, and the derivation tree looks larger, since when it is depicted this way it includes every character of the sentence, in addition to the indicated functions and constituent structure. But in fact there is a very natural grammatical coding scheme in which this derivation is specified by just 5 bits.

We can read off the elements of this tree by "traversing" the tree in a standard order. A standard "LR traversal" yields the derivation in a kind of reverse Polish notation:

every D, student N, f, criticized V, some D, teacher N, f, g, h

Now, we can calculate the size of this derivation by considering how many choices there are in this 9 element LR traversal. Since every sentence in the language begins with some determiner, and there are exactly 2 determiners, the first element represents one binary choice, one bit. After a determiner, we always have a noun, so that is a second binary choice. After a determiner and noun, the 2-place function f invariably applies. There is no choice here, so this is 0 bits. Considering our choices in this way, given each prefix p of the LR sequence, we easily calculate:[4]

$$size = \sum_{LR\ prefixes\ p} log_2|choices(p)| = 5 \text{ bits}$$

Notice that, whenever Lex and $\mathcal{F}$ are finite, the number of choices at each point in the LR enumeration of derivations will be finite. So

[4]As in the input sequence, we are not counting the last choice of whether to continue to build a more complex form or not. Adding that choice into our size measure would not affect the points being made here.

The choice function will not be defined here, but this kind of function is familiar from parsing applications, where an "LR oracle" specifies the choices compatible with prefixes of the input (often focusing on the ideal deterministic case where there is exactly one choice at each point).

in general, a grammatical code of this kind can be provided, and it will tend to be smaller than an input code (assuming that not all strings are grammatical). With any natural language grammar worth having, specifying an expression by its shortest derivation requires, on average, many fewer decisions than specifying it as an arbitrary sequence of sounds or typed characters. We will use this notion of size below.

Grammars like the one given above cannot specify "rolling-up" derivations of the sort mentioned earlier. The structure building rules here are essentially like context free rules, and we will see that these cannot possibly suffice. We now define transformational structure building rules that will do the trick.

16.3 A minimalist grammar formalism

The $\mathcal{MG}$ framework is defined by the restrictions it places on lexicons and generating functions.

16.3.1 Lexicon

An $\mathcal{MG}$ lexicon is a finite set of lexical items, where each lexical item is a finite sequence of features. There are four kinds of syntactic features, and there may also be non-syntactic (e.g. phonetic) features:

`c, t, d, n, v, p,...`	(selected categories)
`=c, =t, =d, =n, =v, =p,...`	(selector features)
`+wh, +case, +focus,...`	(licensors)
`-wh, -case, -focus,...`	(licensees)
`every,some,student,...`	(non-syntactic: phonetic,semantic)

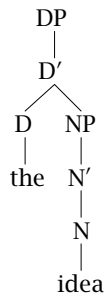
Given the structure building rules of the next section, it will be easy to show that a lexical item that has a non-syntactic feature preceding any syntactic feature will be unable to play a role in any non-trivial derivation, because the structure building rules "check" syntactic features from left-to-right.[5]

[5] Allowing for the possibility of "useless" lexical items in order to obtain a maximally simple account is analogous to allowing "useless" categories to obtain a simple definition of context free grammars, where a category is "useless" if it has no yield in Σ^* or cannot be reached from the start category "S". It is of course possible to rule out such "useless" categories, just as we could rule out "useless" lexical items, but this complicates the definition of the grammar.

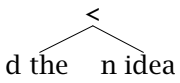
16.3.2 Structure building rules

Our complex expressions will be trees, so it is convenient to regard lexical items as trees too. Lexical items are just 1-node trees, labeled with finite sequences of features.[6] A one node tree is both a root and a leaf.

We will also use a special tree notation which will make it easy to see how the structure building rules are feature-driven. We will not use traditional X-bar structures like



Rather, we will have trees like the following:



The internal nodes of these trees are labeled by the "projection order" of the daughters. In effect, the symbols <, > "point" towards the *head* of the phrase. So we can see that the tree just displayed is a determiner phrase by the fact that the order "points" to the left daughter which has the categorial feature d. Every leaf is a *head*, as usual. A *maximal projection* is a maximal subtree with a given head. So, for example, the right daughter of the tree just displayed is a head but also a maximal projection, since no larger subtree has that node as its head. Any right sister of a head will be called a *complement*. Any phrase attached to the left of a head in its maximal projection will be called a *specifier*.[7]

[6] Allowing complex trees as lexical items, as many linguists have proposed, does not change things substantially with respect to any of the issues considered in this paper.

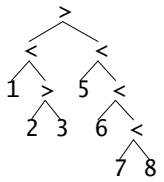
[7] As will become clear below, a constituent can have at most one complement, but for the moment, no restriction is placed on the number of specifiers that a constituent can have.

Let exp be the set of expressions, that is, the set of trees of this kind, in which internal nodes are labeled with < or > and leaves are labeled with (possibly empty) sequences of features.

Let's write t_1COMPt_2 to mean that the head of t_1 has t_2 as its complement (i.e. its right sister). That is, COMP signifies the "has the complement" relation. We let COMP$^+$ be the transitive closure of COMP, and COMP* be the reflexive, transitive closure of COMP. We will say that t_2 is a comp$^+$ of t_1 if t_2 is a complement of, or a complement of a complement of, or ...of t_1.

Similarly, let's write t_1SPECt_2 to mean that the head of t_1 has t_2 as its specifier. That is, SPEC signifies the "has the specifier" relation. We let SPEC$^+$ be the transitive closure of SPEC, and SPEC* be the reflexive, transitive closure of SPEC. We will say that t_2 is a spec* of t_1 if t_2 is t_1, or else it is a specifier of, or a specifier of a specifier of, or ...of t_1.

These relations do not depend on the node labels, but only on the structure of the tree and the < or > ordering. For example, consider the following tree:

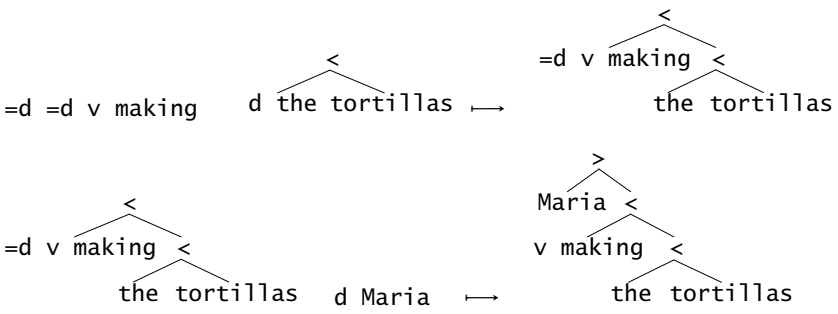


The head of this tree is labeled 5. The constituent $[_<1[_>2,3]]$ is a specifier and hence also a spec$^+$ of 5. The constituent 1 is a specifier of 2, and so is also a spec$^+$ of 5. The constituent $[_<6[_<7,8]]$ is a complement of 5. The constituents $[_<6[_<7,8]]$, $[_<7,8]$ and 8 are comp$^+$'s of 5.

Now we define the two structure building rules: $merge$ and $move$.

The partial function $merge : (exp \times exp) \rightarrow exp$

The $merge$ function applies to a pair of expressions when the head of the first begins with a selection feature =x for some x, and the head of the second begins with that category x. A 1-node head attaches the first constituent it selects on the right, in complement position. If it selects any other constituents, these are attached to the left in specifier positions, as we see in the following examples:



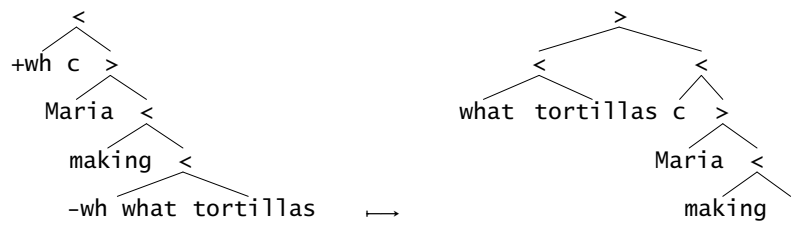
We also see in these examples that this structure building operation "cancels" a pair of features: the selection feature =x and the selected feature x are deleted, and do not appear in the result. In the first example, we see that the input trees have 4 syntactic features altogether, while the result has 2. In the second example, the inputs have 3 syntactic features, and the result has only 1.

We can define this structure building function in the following way, adapted from Cornell (1997a). Let $t[f]$ be the result of prefixing feature f to the sequence of features at the head of t. Then, for all trees t_1, t_2 and all $\mathsf{c} \in Cat$,

$$merge(t_1[\mathsf{=c}], t_2[\mathsf{c}]) = \begin{cases} \begin{array}{c} < \\ t_1 \quad t_2 \end{array} & \text{if } t_1 \in Lex \\ \begin{array}{c} > \\ t_2 \quad t_1 \end{array} & \text{otherwise} \end{cases}$$

The partial function: $move : exp \rightarrow exp$

The function $move$ applies to a single tree, when the head of that tree has a feature +f as its first element and when there is exactly one -f feature in the tree, where that feature is initial in some comp$^+$ of the tree or some specifier of a comp$^+$ of the tree. In this case, move applies by moving the maximal projection of the -f subtree to specifier position, and replacing the original occurrence of that tree by the empty tree λ (i.e. a single leaf node with no features). We see this operation applying in the following example:



Again, it is important to notice that each application of this function cancels exactly two features: the +f and -f are deleted and do not appear in the result. In this example, the input has 3 syntactic features, and the result has only 1.

We can define this structure building function in the following way. Let $t^>$ be the maximal projection of t. And for any trees t, t_1, t_2 where t properly contains t_1 but does not contain t_2, let $t\{t_1/t_2\}$ represent the result of replacing t_1 by t_2 in t. Then for any tree $t_1[\texttt{+f}]$ which contains exactly one exposed occurrence of -f, where $t_2[\texttt{-f}]^>$ is a comp$^+$ or a specifier of a comp$^+$ of $t_1[\texttt{+f}]$:

$$move(t_1[\texttt{+f}]) = \overset{>}{\diagup\diagdown}\ t_2^> \quad t_1\{t_2[\texttt{-f}]^>/\lambda\}$$

With this definition, all movement is overt, phrasal, and leftward.

The requirement that movement cannot apply when two outstanding -f requirements would compete for the same position is a strong version of the "shortest move" condition Chomsky (1995). The other restriction on movement proposed here is from Koopman and Szabolcsi (1998): the moved tree must be a comp$^+$ or the specifier of a comp$^+$.
Koopman and Szabolcsi (1998) say that the extracted constituent must be "on the main projection line," relating this idea to the treatment of the ECP in Kayne (1984).

$\mathcal{MG}$ grammars and languages

To summarize, in $\mathcal{MG}$, each grammar $G = \langle Lex, \mathcal{F}\rangle$ where the lexicon is a finite set $Lex \subseteq$ features*, and where $\mathcal{F} = \{merge, move\}$. As usual, a grammar of this kind defines a set $L(G) = closure(Lex, \mathcal{F})$. Let's say that the *sentences* in such a set are those $s \in L(G)$ with just one occurrence of one syntactic feature left, namely, the "start" category c.

We will use $\mathcal{MG}$ for the set of all minimalist grammars, and $\mathcal{ML}$ for the set of languages definable by these grammars, i.e. the string sets of the sentences of these languages.[8]

16.4 Formal grammars for Hungarian verbal complexes

The $\mathcal{MG}$ grammar formalism easily defines remnant movement. This paper will take three first steps towards the Koopman and Szabolcsi analysis of Hungarian verbal complexes. All three steps share the formal properties which will be explored later, but it is interesting to see how the beginnings of the more sophisticated analysis can be introduced.[9]

16.4.1 $G1$: rolling up VPs

The "rolling-up" of verb phrases schematically indicated in (8) can be defined with a trivial grammar. Let $G1 = \langle Lex, \mathcal{F} \rangle$ where Lex has the 9 lexical items listed here:

Lex:

=v v V1		
=v v V2		=v +v v V2
=v v V3	=v +v v V3	=v +v v -v V3
=m +m v V4	=m +m v -v V4	
m -m M		

$\mathcal{F} = \{merge, move\}$

In this lexicon, I use lower case for syntactic features, and upper case for the non-syntactic features. The first column of entries is used to derive (5), with obligatory inversion of the modifier M triggered by a movement requirement -m. The second column adds entries for (6), so that after M inverts with V4, the result still has a -v feature which forces M V4 to invert with V3. The third column adds entries for (7), so that after M V4 inverts with V3, it still must invert with V2.

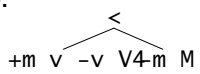
[8] These grammars are weaker than the grammars studied by in previous work by Stabler, Cornell, and others. Cornell (1997a) shows how a derivational presentation of the sort given here, one in which traces and chains play no role, can equivalently be presented representationally, defining the language of expressions with traces, chains and constraints, more along the lines explored by Brody (1995). In other work Cornell (1997b,1998a,1998b) shows how grammars of this sort can be implemented in the type logical framework presented in Moortgat (1996) and Moortgat and Oehrle (1996). These demonstrations go through, even more easily, in the simpler framework described here. See also the related work of Lecomte (1998), Retoré and Lecomte (1997), Becker (1998), Michaelis (1998), Michaelis and Wartena (1998), Szalai and Stabler (1998), Vermaat (1998).

[9] See Szalai (forthcoming) for more sophisticated formal models of these structures.

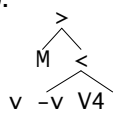
Notice that V3 has three forms: the first just selects v; the second adds the +v feature so that it will invert the VP it selects, and the third adds -v so that its maximal projection must also invert. It is this third form that yields the distinctive, unbounded "rolling up" of structure. Notice that this form introduces two features =v,+v that it also consumes. – It is a common idea from categorial grammar that a category (e.g. a modifier) could yield the same resource that it consumes; what is novel here is that one of the resources with this status is one that triggers movement.

Here are the first steps of a derivation of a VP:

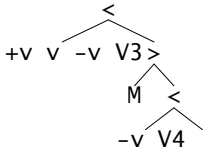
1. lexical: m -m M
2. lexical: =m +m v -v V4
3. merge(2,1):



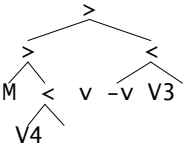
4. move(3):



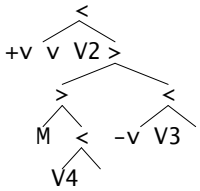
5. lexical: =v +v v -v V3 ⇐ this is the recursively moving category!
6. merge(5,4):



7. move(6):



8. lexical: =v +v v V2
9. merge(8,7): ⇐ next step, VP3 rolls up



In step 5, the recursive category is introduced This first grammar allows derivations that we wanted to exclude, though, like: * V1

V3 M V4 V2. So let's go to a slightly more interesting grammar.

16.4.2 *G*2: introducing PredP

To account for a wider range of facts, we need something a little more sophisticated. Following Koopman and Szabolcsi, we take another step. First, the contents of the position we are calling M are not all alike: they are classified into "bigger" and "smaller" types. A WP with a bigger M in its spec must be selected by PredP; with smaller M, PredP is optional. The modifier *haza* in our Hungarian examples (5)-(7) is "smaller," in this sense. The presence of PredP then determines a later step. When WP is selected by PredP, it cannot extract; the CP above it must extract. That is, PredP (our category `pr`) blocks inversion, and is always present when the modifiers are "bigger."

So in the following schematic representation, we see the derivation of the English form deviate from that of the inverted form after the first two steps. Two steps build a WP, but at that point the WP is optionally selected by PredP. If it is not selected by PredP, we get WP-to-WP movement for the inverted order. And if it is selected by PredP, we get CP-to-WP movement for the English order. The differences between the two derivations occur at the underlined steps.

(9) a. V2 M ⇒ invert M
b. $[_w$ M V2] ⇒ For inverted order: build CP, VP1
c. V1 $[_c$ $[_w$ M V2]] ⇒ WP to spec,WP
d. $[_w$ $[_w$ M V2] V1 $[_c$]] ⇒ CP to licensor
e. $[_c$] $[_w$ $[_w$ M V2] V1]⇒ WP to higher WP or CP
f. $[_w$ $[_w$ M V2] V1] $[_c$]

(10) a. V2 M ⇒ invert M
b. $[_w$ M V2] ⇒ for English order: WP selected by PredP
c. $[_{pr}[_w$ M V2]]⇒ build CP, VP1
d. V1 $[_c$ $[_{pr}[_w$ M V2]]]⇒ CP to spec,WP
e. $[_w$ $[_c$ $[_{pr}[_w$ M V2]]] V1] ⇒ CP to licensor
f. $[_c$ $[_{pr}[_w$ M V2]]] $[_w$ V1] ⇒ WP to higher WP or CP
g. $[_w$ V1] $[_c$ $[_{pr}[_w$ M V2]]]

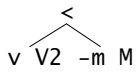
To get just the aspects of the analysis indicated above, and continuing to ignore the arguments of the verbs, we use the following grammar with 12 lexical items. Th grammar introduces the categories `w`, `pr`, and the CP licensing category `lc`. Let $G2 = \langle Lex, \mathcal{F} \rangle$ where,

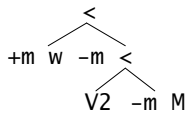
Lex:	=lc +m c	=lc +m c -c
	=w +c lc	
	=c v V1	=pr c -m
	=w c -c	=bw pr
	=v +m w -m	=v +bm bw
	=m v V2	
	m -m M	m -bm M

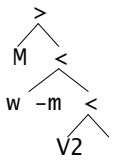
$\mathcal{F} = \{merge, move\}$

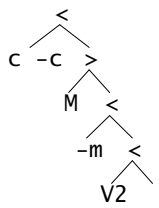
To derive the inverted order M V1 V2, lexical items in the first column suffice. To derive the English order V2 M V1, we add the second column, in which we see the feature -bm – the special requirement of "big modifiers", and the category pr (the category that heads PredP).

The following is a complete derivation of the inverted order M V2 V1 from this simple grammar:

1. lexical: m -m M
2. lexical: =m v V2
3. merge(2,1):


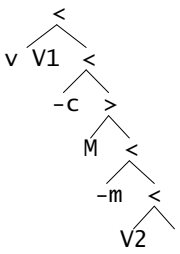
4. lexical: =v +m w -m
5. merge(4,3):


6. move(5):


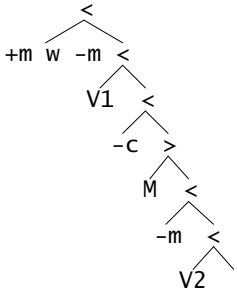
7. lexical: =w c -c
8. merge(7,6):


9. lexical: =c v V1

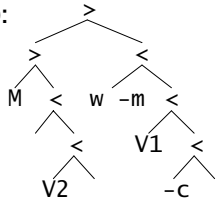
10. merge(9,8):



11. merge(4,10):

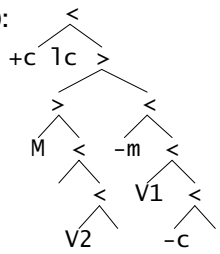


12. move(11): WP to spec,WP

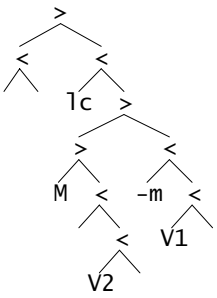


13. lexical: =w +c lc

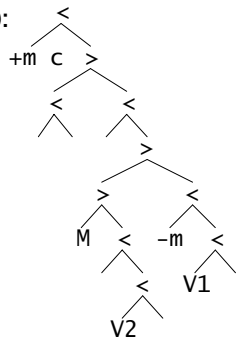
14. merge(13,12):



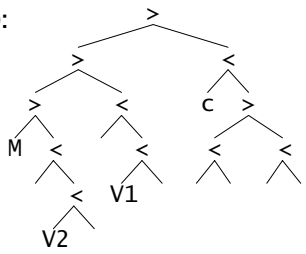
15. move(14): CP to licensor



16. lexical: `=1c +m c`
17. merge(16,15):

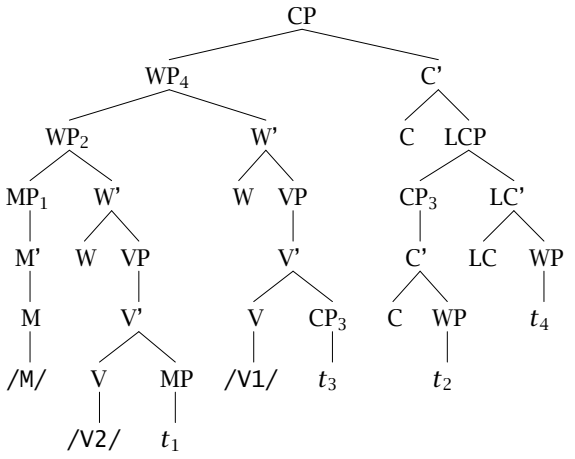


18. move(17):



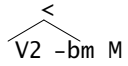
This derivation of the "inverted order" has 18 steps: 7 lexical items and 11 function applications. Notice that the derived structure 18 has only 1 syntactic feature left, namely the start category `c`. All the empty nodes are of little help in remembering how the derivation went, and so it is no surprise that linguists prefer richer labels! The traditional linguistic depiction (11) shows how the derivation went but obscures whether all features were checked:

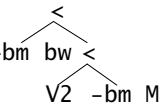
(11)



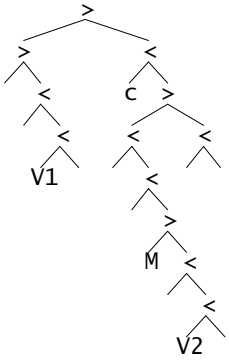
Looking at this tree, it is easy to see the course of the derivation, but it is very hard to see whether the derivation was successful – i.e. whether all features were checked! Having done the derivation we know: they were. (We will consider another, rather different representation of this derived structure below in section 16.7.)

The derivation of the English order `V1 M V2` differs from the previous derivation at the first step and then, as a consequence of that first choice, later in the derivation:

1. `lexical: m -bm M` ⇐ different from prev derivation here!
2. `lexical: =m v V2`
3. `merge(2,1):`


4. `lexical: =v +bm bw`
5. `merge(4,3):`


…

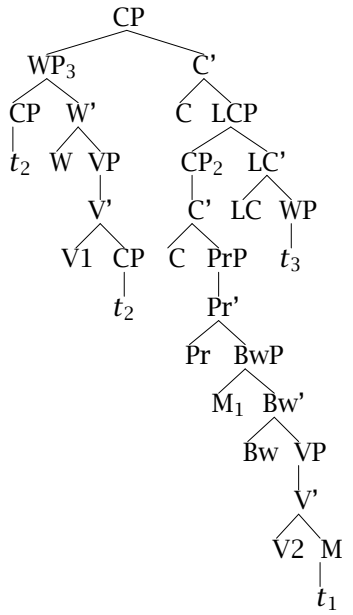
21. `move(20):`


This is the English order

This derivation uses 9 lexical items, 12 function applications. It has 2 more lexical items than the derivation of inverted order, because (i) the inverted order uses `=v +m w -m` twice, while this one uses `=v +bm bw` in one of those positions, and (ii) this derivation has `=bw pr`. No surprise: this derivation has one more function application step than the inverted order, namely: the merging of PredP `=bw pr` with the "big" WP.

The traditional depiction of the derived structure is the following:

(12)



Koopman and Szabolcsi have a more elegant way to force CP movement when there is a PredP, and to allow the checked requirement to be the -m feature of WP rather than a -m that is arbitrarily assigned to CPs with PredPs in them. One further step towards their analysis will be taken in the next section, where the extraction of spec,$\mathtt{WP}_3$ gets some attention.

16.4.3 *G*3 and a conservative extension for pied piping

One of the interesting ideas in the Koopman and Szabolcsi proposal is that we have been overly liberal in our assumptions about what can move and overly conservative in our assumptions about pied piping. They propose that pied piping and the movement of large structures is the normal occurrence. Let's see how we can adopt the following assumptions:

(13) No projection has more than one specifier.

(14) Only complements (that is, comp$^+$s) move.

(15) A +f head triggers the movement of a comp$^+$ to check a -f feature in the spec* of that constituent.

To enforce the first restriction, we require lexical items to have the form:

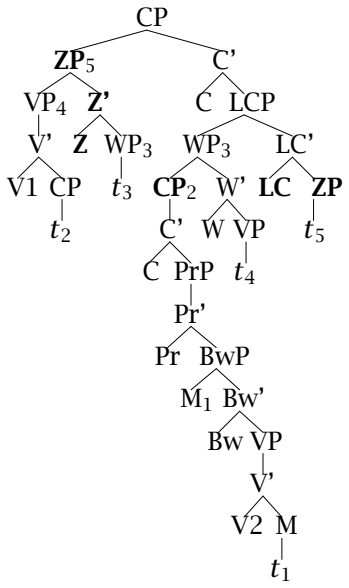
`(=f({+g,=g})) c -r* (phon) sem`

where parentheses indicate optional choices (i.e. 0 or 1 occurrences), braces indicate the selection of exactly 1 element, and the star indicates 0 or more occurrences. (We neglect semantic features `sem` in the present study.)

To enforce the second two restrictions, we simply modify the definition of *move* so that a `+f` head triggers the movement of a comp$^+$ to check a `-f` feature in the spec* of that constituent.

Let $\mathcal{SMG}$ be the set of these "strict minimalist grammars," and let $\mathcal{SML}$ be the set of languages definable by these grammars, i.e. the string sets of the sentences of these languages.

The grammar G2 on page 311 already respects the lexical restriction: at most 1 specifier per head. However, with that grammar some sentences are derived by moving specifiers. For example, the "English order" structure with the derivation ending on page 315 involves specifier movement. The specifier movement is clearly visible when the derived structure is depicted as in the tree 12. To avoid this, the strategy is to redesign the grammar so that the `VP` in comp,`WP`$_3$ moves to some spec,`ZP`, so that then `WP`$_3$ can move to spec,`LCP` to check `-c` in its spec*, leaving us with the same English order of pronounced heads. That is, we will modify the grammar to yield the structure:

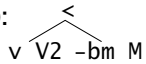


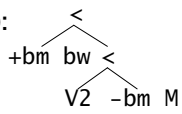
In this structure, all traces are in complement position. To empirically support any analysis of this sort, it of course remains to identify and independently motivate ZP, but the focus here is just on the formal options, so we provide a grammar which just uses the new projection ZP in the way indicated:

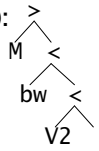
=lc +m c	=lc +m c -c
=z +c lc	
=wz +v z -m	**=v +m wz**
=c v -v V1	=pr c -m
=w c -c	=bw pr
=v +m w -m	=v +bm bw
=m v V2	
m -m M	m -bm M

Notice that category wz is introduced which is not -m, since z is -m instead.

With this grammar, the derivation of English order begins as usual, but gets to step 20, where a specifier triggers the movement of a complement:

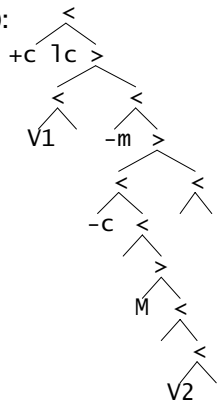
1. lexical: m -bm M
2. lexical: =m v V2
3. merge(2,1):


4. lexical: =v +bm bw
5. merge(4,3):


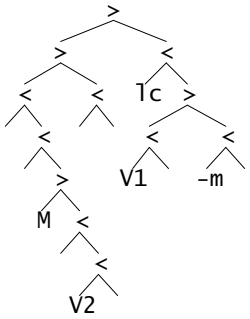
6. move(5):


…

20. merge(19,18): ⇐ the -c phrase cannot move

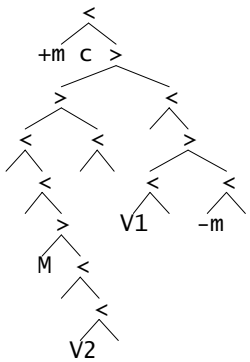


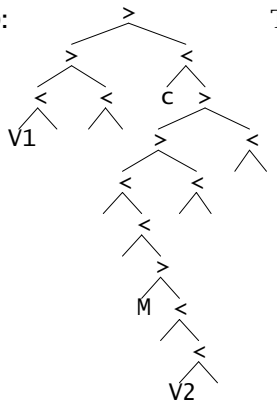
21. move(20):



22. lexical: `=lc +m c`

23. merge(22,21):



24. move(23): 

This is the English order

Notice the movement of the complement to check a feature in its specifier in step 21.

16.5 Remnant movement and expressive power

First, it is easy to see that all context free languages can be defined in the $\mathcal{MG}$ formalism, and that the class of minimalist grammar languages is closed under unions:

(16) $\mathcal{CFL} \subseteq \mathcal{ML}$

(17) If $L1, L2 \in \mathcal{ML}$, $L1 \cup L2 \in \mathcal{ML}$

Stabler (1997) showed that languages with 5 "counting dependencies" could be defined in a transformational grammar formalism with head movement and covert movement. It turns out that even without those mechanisms, we can still define these complex languages, very easily, as we see in the following grammar.

Theorem: $\{1^n 2^n 3^n 4^n 5^n \mid n \in \mathbb{N}\} \in \mathcal{ML}$

```
=1 +5 +4 +3 +2 +1 c        c
=2 1 -1 /1/                =2 +1 1 -1 /1/
=3 2 -2 /2/                =3 +2 2 -2 /2/
=4 3 -3 /3/                =4 +3 3 -3 /3/
=5 4 -4 /4/                =5 +4 4 -4 /4/
   5 -5 /5/                =1 +5 5 -5 /5/
```

It is easy to check that all movements in derivations of clauses from this grammar are movements of complements, but the first lexical item allows a c to have 5 specifiers. The grammar is easily changed to the "strict" $\mathcal{SMG}$ format, just by trading in the one offending lexical item for 5:

```
=1 +5 +4 +3 +2 +1 c  ⇒  =1 +5 c5
                        =c5 +4 c4
                        =c4 +3 c3
                        =c3 +2 c2
                        =c2 +1 c
```

Consequently, we have $\{1^n2^n3^n4^n5^n \mid n \in \mathbb{N}\} \in \mathcal{SML}$. In fact, it is plausible that the restrictions on the "strict" grammars are "conservative" in the sense that they do not change the class of definable languages:
Conjecture: $\mathcal{SML} = \mathcal{ML}$.

Vijay-Shanker and Weir (1994) show that languages with 5 counting dependencies cannot be defined by tree adjoining grammars (TAGs), nor by linear indexed grammars (LIGs), nor by head grammars (HGs), nor by categorial grammars (CGs) of a certain kind. Rambow (1994) points out that these languages can be defined by "multiset-valued linear indexed grammars" ({} − LIGs).

The following conjecture also seems plausible, though it has not yet been proven:

Conjecture: Grammars in $\mathcal{MG}$ can define languages with more than 2 counting dependencies only when some sentences in those languages are derived with remnant movements.

That is, it appears that the possibility of remnant movement is responsible for the great expressiveness of the framework. Prohibiting remnant movement, while leaving the framework unchanged in other respects, would yield a considerably less expressive system.

Many other basic questions remain open:[10]

(18) Open: Is $\mathcal{ML}$ closed under intersection with regular languages?

(19) Open: Is $\mathcal{ML}$ closed under ϵ-free concatenation?

(20) Open: Is $\mathcal{ML}$ closed under ϵ-free homomorphisms?

16.6 The size of verbal complexes

The derived structures of the previous examples may appear to be rather complex, especially when depicted in the form that is familiar from mainstream theoretical linguistics, as for example in the tree (11). But in terms of grammar $G2$ and the size measure defined

[10]Michaelis (1998) shows that minimalist languages are a subset of the multi-component TAG languages, $\mathcal{ML} \subseteq \mathcal{MC}\text{-}\mathcal{TAL}$. If it could also be shown that $\mathcal{MC}\text{-}\mathcal{TAL} \subseteq \mathcal{ML}$, then we would have positive answers to the questions (18-20).

above, it has size: 2 bits. With grammar $G2$ one choice in the derivation of this tree comes in the selection of the verbal modifier. If `m -m M` is selected, then we cannot have a PredP, and we can only get the inverted order. If `m -bm M` is selected, then we must have a PredP, and we can only get English order. The other choice is whether to use the first (recursive) or second (non-recursive) `c`.[11]

16.7 Derivations as proof nets

Recent work on proof net representations of grammatical constructions suggests this much better representation of derivations.[12] The reason that the conventional depiction (11) looks complex is that it is showing too many things at once. All that is required for a successful derivation is that all features are checked in appropriate order, and this is something that can be depicted quite easily. For example, the successful derivation of the inverted form has the structure shown in Figure 1: It is a rooted, directed acyclic graph showing all the lexical items (from top to bottom in argument order, i.e. the order they have not in the derived structure but in the derivation tree), and all the matchings. When "movement links" are ignored, i.e. the arcs connecting features `+f` to `-f`, the result is a tree, a tree of "merge links."

A detailed study of these matching graphs goes beyond the scope of this paper, but clearly, it will not be difficult to check such a graph to make sure that (i) features have been discharged in the required order, (ii) all movements are upward, (iii) all movements have applied to constituents that are either comp$^+$ or spec,comp$^+$, and (iv) no movement has applied to a constituent with two outstanding, matching requirements. Notice that there is no need to check for cycles as is required in the proofnets of the type logical tradition. In effect, we have only "tensor links" since all of our grammatical types are first order and there is no hypothetical reasoning.

[11] Without the first lexical item of our grammar $G2$, the grammar gets no other sentences than the two I have derived here because recursion through CP is then bounded. This can be seen by noticing that `=lc +m c` cannot appear in an embedded position, because `c` is only selected by a `v` which will always be dominated by a `-c` category: either `w` or `pr`. So then, notice that the recursive complementizers are both `-c`. But then recursion must be bounded because the only element that can cancel `-c` is `lc`, and `lc` is only selected by the non-recursive complementizer. If we had another complementizer in the lexicon to allow recursion here, the size of the trees would be 1 bit larger, given the choice of whether to use the recursive category.

[12] Retoré (1996); Lecomte and Retoré (1998); Lecomte (1998); Morrill (1997); Moortgat and Oehrle (1998).

(21)

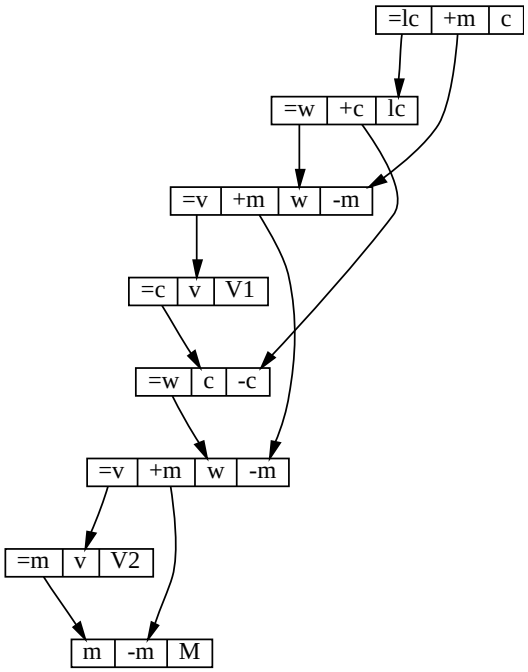


FIGURE 1 Matching graph for the derivation of inverted form

16.8 Conclusions

The generative mechanisms of the "minimalist grammars" formalized here are assumed to be universal, so all language-specific constraints are given in the lexicon. These grammars can define interesting languages very easily, and at least some of the more interesting languages are defined using remnant movement, that is, movements of constituents from which material has already been extracted. Despite the appearance of conventional representations, the real size of the expressions derived by remnant movement can be perfectly reasonable, and the derivations in this framework are naturally represented by (rooted, acyclic) networks of matchings, for which we can provide direct admissibility criteria, phonetic and semantic denotations. Though a fully detailed, formal reconstruction of recent analyses of verbal complexes goes beyond the scope of this paper, the network representation of the basic outlines of the Koopman and Szabolcsi account reveals an extraordinary simplicity that is concealed in conventional representations of the derivational structures.

References

Becker, M. 1998. Acquiring grammars with complex heads. In *Proceedings, Twentieth Annual Meeting of the Cognitive Science Society*.

Brody, M. 1995. *Lexico-Logical Form: A Radically Minimalist Theory*. Cambridge, Massachusetts: MIT Press.

Chomsky, N. 1995. *The Minimalist Program*. Cambridge, Massachusetts: MIT Press.

Cornell, T. L. 1997a. Representational minimalism. SFB 340 Technical Report #83, University of Tübingen. Revised version forthcoming in U. Mönnich and H.-P. Kolb, eds.

Cornell, T. L. 1997b. A type logical perspective on minimalist derivations. In *Proceedings, Formal Grammar'97*. Aix-en-Provence.

Cornell, T. L. 1998a. A deductive calculus of categories and transformations. Forthcoming.

Cornell, T. L. 1998b. Island effects in type logical approaches to the minimalist program. In *Proceedings of the Joint Conference on Formal Grammar, Head-Driven Phrase Structure Grammar, and Categorial Grammar, FHCG-98*, pages 279–288. Saarbrücken.

De Kuthy, C. and W. D. Meurers. 1998. Incomplete category fronting in German without remnant movement. In *Konferenz zur Verarbeitung natürlicher Sprache, KONVENS'98*.

den Besten, H. and G. Webelhuth. 1990. Stranding. In G. Grewendorf and W. Sternefeld, eds., *Scrambling and Barriers*. NY: Academic Press.

Fanselow, G. 1987. *Konfigurationalität. Untersuchungen zur Universalgrammatik am Beispiel des Deutschen*. Tübingen: Gunther Narr Verlag.

Kayne, R. 1984. *Connectedness and Binary Branching*. Dordrecht: Foris.

Kayne, R. 1994. *The Antisymmetry of Syntax*. Cambridge, Massachusetts: MIT Press.

Kayne, R. 1998. Overt vs. covert movment. *Syntax* 1(2):128–191.

Keenan, E. L. and E. P. Stabler. 1996. Abstract syntax. In A.-M. DiSciullo, ed., *Configurations: Essays on Structure and Interpretation*, pages 329–344. Somerville, Massachusetts: Cascadilla Press.

Keenan, E. L. and E. P. Stabler. 1997. Syntactic invariants. In *6th Annual Conference on Language, Logic and Computation*. Stanford.

Kenesei, I. 1989. Logikus – e a magyar szórend? *Általános Nyelvézeti Tanulmányok* 17:105–152.

Koopman, H. 1996. The spec-head configuration. Syntax at Sunset: UCLA Working Papers in Syntax and Semantics, edited by Edward Garrett and Felicia Lee.

Koopman, H. and A. Szabolcsi. 1998. *Verbal Complexes*. UCLA. Forthcoming.

Koster, J. 1994. Predicate incorporation and the word order of Dutch. In G. Cinque, J. Koster, J.-Y. Pollock, L. Rizzi, and R. Zanuttini, eds., *Paths Towards Universal Grammar: Studies in Honor of Richard S. Kayne*, pages 255–276. Washington, D.C: Georgetown University Press.

Lecomte, A. 1998. Pom-nets and minimalism. In *Proceedings of the Roma Workshop*. INRIA, Nancy.

Lecomte, A. and C. Retoré. 1998. Words as modules. In C. Martin-Vide, ed., *Proceedings, International Conference on Mathematical Linguistics, Tarragonna, 1996*. Amsterdam: John Benjamins.

Michaelis, J. 1998. Derivational minimalism is mildly context-sensitive. In *Proceedings, Logical Aspects of Computational Linguistics, LACL'98*. Grenoble. Available at http://www.ling.uni-potsdam.de/ michael/papers.html.

Michaelis, J. and C. Wartena. 1998. Unidirectional inheritance of indices: A weakly context free facet of LIGs. In *Proceedings of the Joint Conference on Formal Grammar, Head-Driven Phrase Structure Grammar, and Categorial Grammar, FHCG-98*, pages 258–267. Saarbrücken.

Moortgat, M. 1996. Categorial type logics. In J. van Benthem and A. ter Meulen, eds., *Handbook of Logic and Language*. Amsterdam: Elsevier.

Moortgat, M. and R. T. Oehrle. 1996. Structural abstractions. In *Proofs and Linguistic Categories: Applications of Logic to the Analysis and Implementation of Natural Language Theory: Proceedings 1996 Roma Workshop*.

Moortgat, M. and R. T. Oehrle. 1998. Proof nets for structured resources. Utrecht Institute of Linguistics. Forthcoming.

Morrill, G. V. 1997. Geometry of language. Universitat Politècnica de Catalunya.

Müller, G. 1996. Incomplete category fronting. SfS report 01-96, Seminar für Sprachwissenschaft, Universität Tübingen.

Nkemnji, M. A. 1995. *Heavy Pied-Piping in Nweh*. Ph.D. thesis, University of California, Los Angeles.

Rambow, O. 1994. *Formal and computational aspects of natural language syntax*. Ph.D. thesis, University of Pennsylvania. Computer and Information Science Technical report MS-CIS-94-52 (LINC LAB 278).

Retoré, C. 1996. Pomset-logic: a non-commutative extension of classical linear logic. In J. Hindley and P. de Groote, eds., *Typed Lambda-Calculus and Applications, TLCA'97*, pages 300–318. NY: Springer-Verlag (Lecture Notes in Computer Science 1210).

Retoré, C. and A. Lecomte. 1997. Logique des ressources et reseaux syntaxiques. In D. Genthial, ed., *Proceedings, Traitement Automatique du Langage Naturel, TALN'97*, pages 70–83. Grenoble.

Stabler, E. P. 1997. Derivational minimalism. In C. Retoré, ed., *Logical Aspects of Computational Linguistics*, pages 68–95. NY: Springer-Verlag (Lecture Notes in Computer Science 1328).

Szalai, T. A. 1998. Approaching a formal model of Hungarian. UCLA M.A. thesis.

Szalai, T. A. and E. Stabler. 1998. Formal analyses of the Hungarian verbal complex. In *TAG+ Workshop, Institute for Research in Cognitive Science, University of Pennsylvania*.

Vermaat, W. 1998. Derivational minimalism and MMCG. Universiteit Utrecht.

Vijay-Shanker, K. and D. Weir. 1994. The equivalence of four extensions of context free grammar formalisms. *Mathematical Systems Theory* 27:511–545.

Webelhuth, G., ed. 1992. *Principles and Parameters of Syntactic Saturation*. Oxford: Oxford University Press.

Zwart, J.-W. 1994. Dutch is head-initial. *The Linguistic Review* 11:377–406.

Zwart, J.-W. 1997. *Morphosyntax of verb movement: A minimalist approach to the syntax of Dutch*. Dordrecht: Kluwer.