

# Recursive Linearization

Joost Kremers  
University of Nijmegen, The Netherlands  
j.kremers@let.kun.nl

18 July 2000

## 1 Introduction

An important discussion in linguistic research concerns itself with the question how word order can be accounted for in principled ways. In generative syntax, the work of Kayne (1994) has been of great influence to this debate. The basic question of this work is how a hierarchical structure can be translated into a linear structure.

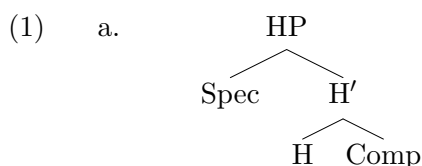
This question, the question of linearization, will also be the main topic of this paper. I will, however, propose a different method of deriving linear order. This method is based on the setting of two parameters for each lexical category. The method put forward here has the ability to derive word order in a rather straightforward way, and has the additional advantage that the hierarchical structure that is linearized, can remain relatively simple. Furthermore, if the procedure outlined here can be developed further, it may even shed some new light on head movement and on the relation between syntax and morphology.

### 1.1 Kayne's LCA

Chomsky (1995; 1998; 1999) argues that syntax is in fact the interplay of two interfaces: the LF interface and the PF interface. As he states in *The Minimalist Program*, (1995), the LF of a linguistic expression should be considered as a purely hierarchical structure. It does not define precedence relations, but solely hierarchical relations. These hierarchical relations will be translated by the interpretational system into particular semantic operations.

In Chomsky (1999), it becomes clear that the PF level is to be regarded as a level that is derived from an LF representation, albeit often an intermediate LF representation. As soon as this position is taken, a principled way of deriving the linear PF structure from the hierarchical LF structure must be provided. The proposal made in Kayne (1994) gives one method of doing this.

Kayne assumes that a linear order is derived from a hierarchical structure in the following way: 'if X asymmetrically c-commands Y, then x precedes y'. Here, X and Y are non-terminal unary nodes dominating the terminal nodes x and y. In this way, a strict ordering is obtained, in which a tree structure as in (1a) will always be linearized as in (1b):



b. Spec H Comp

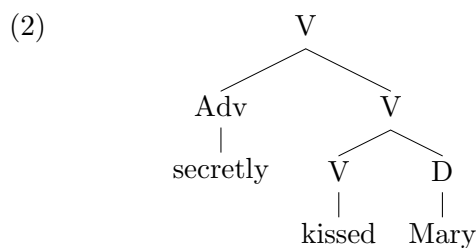
Furthermore, the proposal made by Kayne entails that a head can only have one specifier, and that adjunction is impossible. *is that true, and how exactly?* For this reason, analyses that adopt Kayne’s proposals, usually show a proliferation of functional projections. In order to account for the wide variation in word order that exists across languages, a kaynian analysis needs to propose many movements. However, since a head can only have one specifier and adjunction is not possible, each movement requires a functional head that provides a specifier position that can be targeted.

Unfortunately, a kaynian analysis often needs more landing positions than there are functional projections. For this reason, undesignated functional projections are often assumed, usually indicated with FP, or currently also with WP (for Word-order Phrase). Obviously, the need to posit such a large number of unspecified functional projections is a serious problem for the framework. A functional projection has its place in a tree structure because it corresponds to an actual morpheme on a substantive lexical head, or to an actual freestanding phonological form. However, there does not seem to be any motivation for these FPs or WPs beyond the need to have landing sites for movement.

## 1.2 Minimalist tree structures

Kaynian analyses need to maintain the X-bar schema as it has been used in linguistics since (when?). In his recent work, however, Chomsky makes an attempt to do away with this framework and to replace it with a more basic notion, that of merge. Merge is an operation that takes two elements A and B and puts them together to form a (more) complex structure K(A,B). The new element K gets its label from either A or B. The element that passes its label on to the new node is the element that projects.

In principle, this is all that merge does. Therefore, it cannot distinguish between different layers of a projection. As a result, there is no principled way to ban the merger of more than one specifier, and more fundamentally even, there is no way to distinguish an adjunct from a specifier.<sup>1</sup> Take, for example, the following structure:<sup>2</sup>



In (2), the head V is first merged with a D, which expresses the object. The resulting structure has the label of the projecting node, V. Next, it is merged with an adverb, a merger that is traditionally termed adjunction. The resulting structure is again a V.

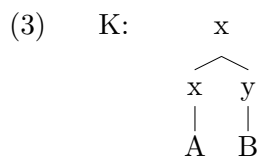
The structures that are formed by merge are not X-bar structures in the traditional way. In a kaynian analysis, the structure in (2) would have to be considered a VP. But since it contains an adverb, this is not desirable. We would like to say that the adverb is at best

<sup>1</sup>See Kremers (in preparation) for a way to treat adjuncts and arguments in a similar way syntactically, without losing the semantic distinction between them.

<sup>2</sup>For ease of exposition, I will abstract away from *v* for the moment.

adjoined to the VP, not a part of it. In a minimalist framework, however, there is no such thing as a VP anymore.<sup>3</sup> The structure in (2) is a V, as much as [<sub>V</sub> kissed [<sub>D</sub> Mary ] ] is a V. Both phrases have the same combinatorial properties. They can both be taken as a complement by T, for example.

On an abstract level, we can say the following. Any node can be described as a structure of the type  $K(A,B)$ , where  $K$  is a compound structure formed out of the elements  $A$  and  $B$ .  $A$  and  $B$  can be terminal elements (heads) or they can be compound structures themselves. Furthermore,  $K$  has a label, which is either the label of  $A$  or the label of  $B$ . I will assume that the label of an element is in fact its category (i.e., N, V, D, T, C, etc.)<sup>4</sup> Thus, the full representation of  $K$  will be  $K_x(A_x, B_y)$ . This represents the following tree structure:

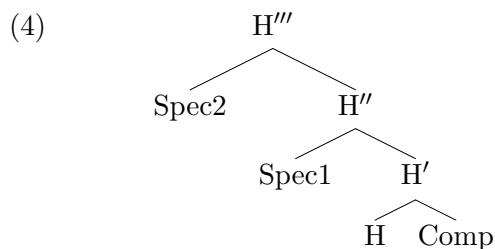


Note that this structure contains hierarchical and categorial information, plus information about the nodes  $A$  and  $B$  that is available at this level: their label, and whether  $A$  or  $B$  is a head.<sup>5</sup> It does *not*, however, contain any precedence information. Precedence information is to be read off the structure.

## 2 The principles of recursive linearization

In this section, I will introduce the principles of the linearization procedure RLin that I propose. Linearizing a node  $K(A,B)$  simply means spelling out  $A$  before  $B$ , or  $B$  before  $A$ . But note that  $A$  and  $B$  may themselves be nodes of the form  $K'(A', B')$ . That means that before spelling out  $A$  and  $B$ , their sub-nodes must first be linearized and spelled out. In this sense, the linearization procedure is recursive. (Hence RLin). Linearizing a node  $K$  is simply deciding in which order  $K$ 's sub-nodes should be linearized.

For ease of exposition, I will refer all through this section to the following abstract tree:



<sup>3</sup>The label VP may still be used at times, but then it is purely a mnemonic device for the highest projection that is still of category V.

<sup>4</sup>Chomsky (1999) hints at the possibility of building trees without labels. I am not sure if that is at all a possibility, but if it is, the present proposal should be easily transferable into such a system.

<sup>5</sup>Note that it is perfectly possible that both  $A$  and  $B$  are heads. For example,  $K$  could have the structure  $K_D(\text{the}_D, \text{man}_N)$ , representing the DP 'the man'. In other words, it is not necessary that ' $\text{man}_N$ ' projects to a full NP before 'the<sub>D</sub>' can take it as a complement. After all, NP is simply the highest projection that is still of the category N, which in this case is ' $\text{man}_N$ '.

I have marked the different levels of  $H$  with primes, but this should not be taken to mean that the computational system can actually distinguish between them. It is solely for ease of reference. The same goes for the terms Spec and Comp. (4) is simply an instance of a head  $H$  that has been merged with three different elements.

When linearizing a node  $K_x(A_x, B_y)$ , there are basically two options. Either spell out  $K$  in the order  $A$ - $B$ , or spell it out in the order  $B$ - $A$ . The question then becomes how we can decide between the two. In fact, there are two pieces of information that can be used for this. First, one can look at the label of  $K$  and of the sub-nodes. We now have two options: we can first spell out either the sub-node with the same label as  $K$ , or the sub-node with a label different from  $K$ . In other words, we can decide whether to spell out the projecting node first or second.

The first strategy, which I call ‘label-first’, will result in the following string:

(5)  $H$  Comp Spec1 Spec2

To see how this works, follow the tree in (4). RLin will start working on  $H'''$ . This node consists of Spec2 and  $H''$ . Remember that the primes are not really there, so  $H''$  is in fact the same label as  $H'''$ . Consequently, following the principle of label-first, RLin will linearize  $H''$  first. This node consists of two sub-nodes, Spec1 and  $H'$ . Here, the same thing happens.  $H'$  is linearized first.  $H'$  consists of two sub-nodes,  $H$  and Comp. Again, following label-first,  $H$  is taken first. But since  $H$  is a terminal node, it can be spelled out.

By spelling out  $H$ , RLin has completed the first part of linearizing  $H'$ . The second part consists of linearizing Comp. As a result, everything in Comp will be spelled out immediately after  $H$ . That completes the linearization of  $H'$ . Next,  $H''$  has to be completed. One node,  $H'$ , has already been done, now the other node is up. Consequently, everything in Spec1 will be spelled out next. And lastly,  $H'''$  will be completed, by spelling out Spec2. In this way, the order in (5) is obtained.

There is, however, another strategy that can be followed. Instead of linearizing the projecting node first (label-first), the non-projecting node can be linearized first, a strategy that can be termed ‘label-second’. Now, when RLin starts with  $H'''$ , it will first take Spec2, before going down into the tree and spelling out the projecting node  $H$ . The reader can check for himself that this strategy will result in exactly the opposite order of (5):

(6) Spec2 Spec1 Comp  $H$

This, however, is not the whole story. As I said, there is a second piece of information in a node  $K(A, B)$  that can be used. It is also known at the level of  $K$  whether the projecting node is a head or not. It is possible to use this piece of information to modify the two strategies just proposed. First, we take the label-first strategy. A modified label-first strategy would be to say that the sub-node with the same label is linearized first, except when this sub-node is a head. That would result in the following order:

(7) Comp  $H$  Spec1 Spec2

Alternatively, one could modify the label-second strategy. In principle the non-projecting sub-node is linearized first, except when the projecting sub-node is a head. That gives the following order:

(8) Spec2 Spec1 H Comp

Given the minimalist tree structure outlined in the previous section, these four orderings are all that are available. All that is required to obtain the proper linearization is two parameters: one ‘label-first (yes/no)’ and one ‘modified (yes/no)’. By setting these two parameters, all four linearization strategies can be described. The full power of the system is only obtained when it becomes possible to set these two parameters for each category independently. That is, a specific language may have a C that is modified label-second, whereas it has a T that is label-first.<sup>6</sup>

### 3 Some examples

So far, the system is very abstract, and it actually yields orders that seem unlikely to be instantiated. In this section, I will discuss three examples that will hopefully clarify the proposal and show its power.

#### 3.1 Arabic noun phrases

Shlonsky (2000) and Fassi Fehri (1999) both make an attempt to present an analysis of Semitic noun phrases. Both authors use an essentially kaynian analysis, although especially Fassi Fehri tries to maintain a number of principles developed by Chomsky, mainly where it concerns move and attract. In this section, I will make an initial attempt at presenting an analysis that is based on a strict application of chomskian principles plus the linearization procedure RLin outlined above.

One of the major insights of the two papers is that in the Arabic noun phrase, modifiers can both precede and follow the noun, but modifiers that precede the head are always heads that do not agree with the noun, whereas the modifiers that follow are always full phrases that do show agreement.<sup>7</sup> Take, for example, the following phrases:

- (9) a. *tālīt-u marra-t-in*  
third.m-NOM time-f-GEN  
‘the third time’  
b. *al-marra-t-u -l-tālīta-t-u*  
the-time-f-NOM the-third-f-NOM  
‘the third time’

(9) shows that ordinals can in principle occur both preceding the noun and following it. When they precede, however, they always take the masculine form, whereas when they follow, they agree with the noun in gender, case and definiteness. The same is true for adjectives. Although they usually occur post-nominally, Fassi Fehri (1999) provides examples which show that adjectives can also productively appear before the noun, taking the noun as a genitive complement, in the same way as the numeral in (9) does:<sup>8</sup>

<sup>6</sup>The scope of this paper does not permit me to go into the possibility that there are certain conditions that apply to the setting of these parameters for different categories. For example, if there is indeed a correlation between C and D, as often assumed, it may be the case that C and D must have the same parameter setting. Other correlations are not excluded.

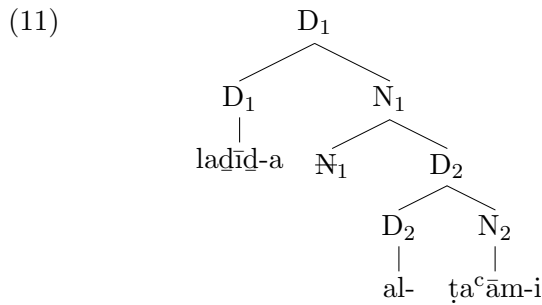
<sup>7</sup>Demonstratives, however, behave differently in Arabic.

<sup>8</sup>The examples are from Fassi Fehri. He admits that the first case, in which the adjective takes the noun

- (10) a. 'akaltu *ladīd*-a          -l-*ṭa<sup>c</sup>ām*-i  
 I ate    delicious-ACC the-food-GEN  
 'I ate the delicious food'
- b. 'akaltu -l-*ṭa<sup>c</sup>ām*-a      -l-*ladīd*-a  
 I ate    the-food-ACC the-delicious-ACC  
 'I ate the delicious food'

Again, the post-nominal modifier agrees with the noun in gender, case and definiteness, whereas the prenominal one does not. Shlonsky (2000) claims that the prenominal modifiers are in fact heads that take the noun as complement. The post-nominal modifiers are in his analysis XPs that occupy specifiers of FPs. The noun is generated below them. A number of XP and remnant movements then result in the surface order.

However, the system of recursive linearization as outlined above is capable of handling the data without needing to resort to movement. Assume for the moment the simplest structure for the noun phrase *ladīd-a -l-ṭa<sup>c</sup>ām-i* in (10a):



In (11), the head *ladīd* has moved from N to D. This is the standard assumption stemming from Ritter (1991). If we now assume that both D and N in Arabic are label-first, they will linearize as exemplified above in (5). The head is linearized first, and the complement follows:

- (12)  $D_1$ - $N_1$   $N_1$   $D_2$   $N_2$

The linearization works as follows: RLin starts to work on the top node. Label-first results in the projecting node,  $D_1$ , being spelled out first. RLin then moves on to  $N_1$ , where the trace of  $N_1$  is spelled out.<sup>9</sup> Next,  $D_2$ , the complement of  $N_1$ , is spelled out. This takes place in the same way: first the projecting node is spelled out, then the non-projecting node. The result is the linear order in (12), which corresponds to the actual order in (10a).

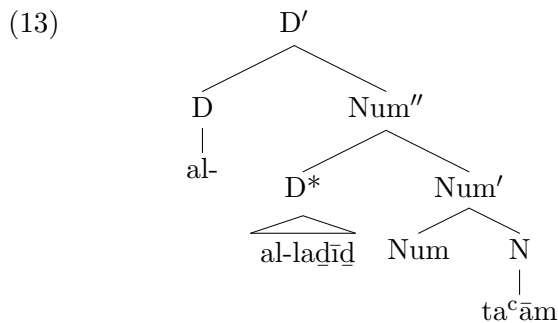
The order of (10b) is derived similarly, provided an additional assumption is made. It must be assumed that there is a head between D and N that takes N as its complement. The presence of such a head is widely argued for, and is generally seen as the noun-phrase equivalent of T. Following Ritter (1991), I will call this head Num, and assume that it carries the number of the noun. Its exact nature is not relevant to the present discussion, however.

Given the assumption of the presence of Num, the tree structure of (10b) is as represented in (13). The adjective is merged with Num, after Num has been merged with N; or, in traditional terms, the adjective is a specifier/adjunct of Num:<sup>10</sup>

as a genitive modifier, is basically a partitive structure, but he claims that in practice, the two structures do not really differ in meaning.

<sup>9</sup>Whatever spelling out of a trace means. I have indicated it here for convenience.

<sup>10</sup>Again, the primes are added for expository convenience only.



Note that I have termed the adjective phrase a D. In this, I follow Fassi Fehri (1999), who assumes that adjectives in Arabic are DPs, for the simple fact that adjective morphology and noun morphology in Arabic are identical.<sup>11</sup>

If one would use the LCA to linearize (13), a problem would occur. Because the adjective is generated as a specifier of Num, it would be linearized between D and N, which is not correct for Arabic. For this reason, additional movements have to be assumed. However, with the recursive linearization and the assumption that D and N, and now also Num, are label-first, the correct order is derived from the tree in (13), without any further movement.

To see how this works, we can track RLin in its path. First,  $D'$  is linearized label-first. That means that the head D, consisting of *al-*, is spelled out. Next,  $Num''$  is linearized, also label-first. That means that spelling out of  $D^*$  is postponed, and that the projecting node  $Num'$  is processed first. This leads to the spelling out of  $Num$ <sup>12</sup>, followed by the spelling out of N. Then RLin returns to  $Num''$  to spell out its second sub-node. The resulting order is the following:

(14) D Num N  $D^*$

This is the order of (10b).

### 3.2 Dutch clause structure

The next example I would like to discuss is that of V2 effects in Dutch main clauses. The effect, by the way, is not restricted to that language. German and other Germanic languages exhibit similar phenomena.

It is a well-known fact that the word order in Dutch main clauses and embedded clauses is not the same. A subclause has a verb-final word order, whereas in a main clause, the finite verb moves to the second position in the sentence, whereby the first position is occupied by the topic. The original explanation of this phenomenon was that in main clauses, V moves to the C position, and that the topic then moves to Spec,CP. However, the assumption of an Infl node in the sentence as the complement of C has made the V2 effect particularly hard to explain.

Given the fact that non-compound tenses in Dutch (and other Germanic languages), tense is expressed on the verb stem, which means that V apparently moves to T in these cases. But if T is the complement of C, it is unclear why the verb is clause-final in subclauses whereas

<sup>11</sup>The star is my addition. I do not mean anything with it, it is merely added to distinguish it from the highest D.

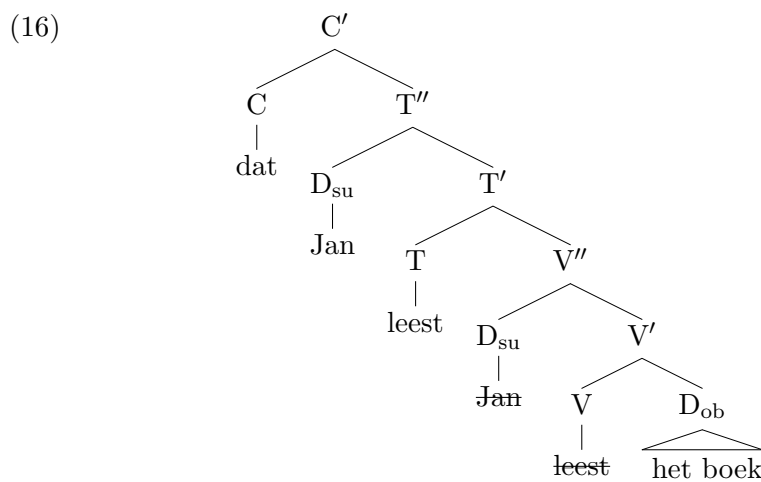
<sup>12</sup>It may be that N has moved to Num, as Ritter (1991) assumes.

there is a V2 effect in main clauses. It would seem that in subclauses, V to T movement does not take place.

In a kaynian framework, the phenomenon seems even harder to explain. However, when one applies the system of recursive linearization, it turns out that the original assumption can be made to work quite easily. Take a typical Dutch subclause, as in (15), with the order complementizer - subject - object - verb:

- (15) ...dat Jan het boek leest  
 ...that John the book reads

Take the tree structure of (15) to be the standard one.<sup>13</sup> Note again that the primes are just added to distinguish the different levels of a projection for the reader. They are not really present in the structure.



Recursively linearizing the tree structure in (16) will give the correct order on the assumption that T and V are label-second, and that C is modified label-second. Let us go through the linearization step by step. First, the top C' node will be linearized. C is modified label-second. That means that the projecting node will be linearized second, unless it is the head. In this case, it is: the projecting sub-node of the top C' node is the head C. Consequently, C is linearized first.

Next, T'', the sister of C, is linearized. Since T is label-second, RLin will spell out the D<sub>su</sub> node first. Then, the T' node is linearized. Again, because T is label-second, V'' will be linearized first. And here the same happens. Because V is label-second, the trace of the subject will be linearized first, and after that V'. Here again, the non-projecting node, D<sub>ob</sub>, is linearized first, and then the projecting node V. At this point, RLin will go back and linearize the remaining node T.

The resulting structure is the following:

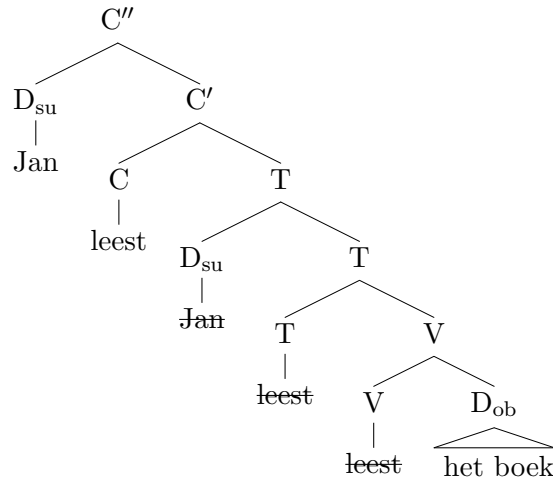
- (17) C D<sub>su</sub> D<sub>su</sub> D<sub>ob</sub> V V T

This is the structure of the Dutch subclause, as exemplified in (15).

<sup>13</sup>I have left out *v* for convenience. The presence of this head would complicate the derivation somewhat. For example, if the object moves and merges with *v*, we must assume a system of mirrored specifiers to derive the right order.

When we now look at main clauses, we see that the same analysis can derive their structure, on the original assumption that Dutch main clauses have a C head that attracts T and which requires that some other element be merged with it.<sup>14</sup> An example is given in (18a), with its tree structure in (18b):

- (18) a. Jan leest het boek  
 John reads the book  
 b.



The linearization of this tree should by now be straightforward. One thing is new, though. The top  $C''$  node is, as assumed, modified label-second. That means that at the highest level, the D sub-node is spelled out first, before the projecting sub-node  $C'$ . But at the level of that sub-node  $C'$ , we find that the projecting node is the head C. That means that, as in (15), the C head, containing the verb *leest*, is spelled out first. The rest of the tree is identical to the subclause, so that the resulting order is as in (19):

- (19)  $D_{su}$  C-T-V  $D_{su}$   $D_{ob}$  V  $\bar{F}$

This is indeed the order of (18a).

### 3.3 English clause structure

After this short discussion of Dutch clause structure, I would now like to take a quick look at English clause structure. English clause structure has a well-known order. In fact, the order of the English clause has more or less formed the ideas in generative linguistics on what the underlying universal order could be.

I argue here, however, that there is no such thing as an underlying word order, let alone a *universal* underlying word order. Rather, the underlying structure of a linguistic expression is a hierarchical one. Now let us see how we can derive the word order of English from it.

English clauses generally show the following word order:

- (20) SU T-V OB

Assuming that SU is the specifier of T and that OB is the complement of V, it is immediately obvious what derives the linear order in (20). The English clause is a typical example of the

<sup>14</sup>With the exception of yes/no questions, which begin with the verb.

principle in (8): modified label-second. All relevant categories, C, T and V, are modified label-second in English.

I will quickly run through the linearization, taking the tree in (18b) as reference. The hierarchical structure of English and Dutch only differs in the fact that English presumably has no C head in main clauses.<sup>15</sup> RLin will therefore start working on the highest T node. D<sub>su</sub> will be linearized first, since it is the non-projecting node, and its sister is not a head. Then RLin moves on to the sub-node T. Now, since the projecting sub-node there is a head, T will be linearized. After this, RLin will continue with the highest V node. In this node, it is the head V that is linearized first, followed finally by D<sub>ob</sub>. The resulting order is that in (20).

The observant reader will have noticed by now that the four linearization principles simply yield the four orders in which an X-bar schema can be linearized. If we take the clause as a model, the following correlations can be observed:

label-first:	H Comp Spec1 Spec2	VOS
label-second:	Spec2 Spec1 Comp H	SOV
mod. label-first:	Comp H Spec1 Spec2	OVS
mod. label-second:	Spec2 Spec1 H Comp	SVO

Table 1: Linearization principles

In the examples discussed in this section, I have used three of the ordering principles in this table: label-first, label-second and modified label-second. It is of course far too early to claim that the fourth principle, modified label-first, does not occur, but it is suggestive that this principle corresponds to a clause structure that, to my knowledge, is not attested as a basic word order in natural language.

If it is indeed excluded, and I must emphasize that it is far too early to make such a statement with any kind of certainty, then it might be the explanation for the fact that not all expected mirror images are found in language. For example, although there are languages with a V2 effect, there are no languages attested that have the mirror image of this effect, what one might call V-penultimate. V-penultimate would require a C that is modified label-first.

There would of course have to be some deeper reason behind the absence of modified label-first. One could tentatively say the following. The basic two linearization principles are label-first and label-second. Label-second has the result of putting the head of a projection at the end. Since the head is in fact the nucleus of a projection, it stands to reason that in certain instances it is preferred not to do that. The principle of label-second is therefore relaxed in some cases, yielding modified label-second.

A similar strategy would not make any sense in the case of label-first, however. Label-first has the result of putting the head first, and there is no reason why one would deviate from this principle. In other words, the term ‘modified’ may not be very accurate. Perhaps the term ‘relaxed label-second’ would be better.

But again, all of this is highly speculative. Empirical research would have to make clear that modified label-first is indeed not found in human language before any of these speculations can be taken seriously.

---

<sup>15</sup>Even if it had, the resulting linear order would be the same. We would have to explain, however, how it is possible that only the subject can move to Spec,CP.

## 4 Some additional remarks

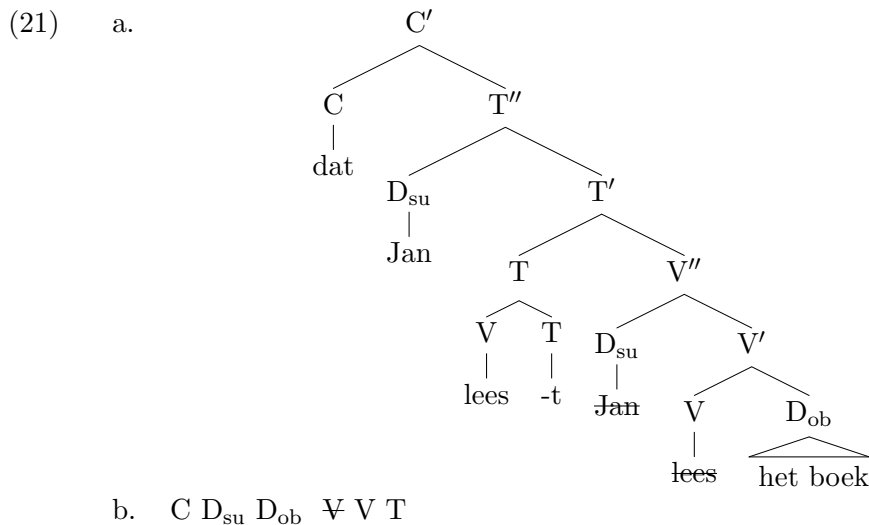
In all three examples given here, Arabic noun phrases and Dutch and English clause structure, I have ignored many questions and problems. It has not been my goal to provide a full account of any of them, I have merely attempted to show that an analysis that uses minimalist tree structures and a linearization procedure such as RLin can in principle do the job. Furthermore, I believe that the analysis proposed here may in fact shed some light on head movement. I will discuss this in the next section, and after that I will finish with some general remarks on linearization parameters.

### 4.1 Head movement

Head movement, as is generally assumed (references?), comes in two types. There is adjunction, in which two heads both have morphological material, which is combined into one word, and there is substitution, in which one head takes the place of another.

The Dutch sentences in the previous section show examples of both types of head movement. The movement of V to T is an example of adjunction: the verb stem is combined with a tense/person suffix. The movement of T-V to C in the main clause is an example of substitution: the head C is presumably an empty element, which is replaced by the lexical material of T-V.

Now look again at the structure of the Dutch subclause:



In the tree structure (21a), I have indicated the movement and adjunction of V to T. In the linear structure (21b), we see that the trace of V actually ends up adjacent to V itself. Although the movement of V puts it into a very different position in the tree structure, in the linear structure, the movement does not have any real effect.

This, of course, is an interesting observation. It opens up the possibility that head adjunction is in fact merely linear head adjacency. It is well possible that the fusion of the morphological material of V and T is the result of the fact that the two heads end up adjacent in the linear structure, not of movement of the lower head to the higher one.

If true, this would be a very appealing analysis. Until now, generative syntax could merely “explain” V to T movement by assuming that T had to be affixal. That is of course

an observation, not an explanation. The explanation offered by morphology has always been an historical one: presumably, tense and agreement markers started out as independent words. Their frequent use with and adjacency to verb stems resulted in unstressing, cliticization and eventually fusion of these elements. The current proposal not only supports that explanation, it provides a syntactic basis for it.

Substitution will be different from adjunction, however. C and T are not adjacent in Dutch. We can tentatively assume that substitution is real movement. However, something more needs to be said about it. If T moves to C, but V has not actually moved to T, how is it then that the verb stem ends up in C? A possible answer could be that T is attracted by C, and when spell-out takes place, it is found that there is no separate T element. T can only be spelled out on a verb stem. As a result, the verb stem is spelled out with T in V2 position.

However, no more morphological material of the verbal complex is dragged along with T than is necessary. Dutch has a type of verb, called ‘separable compounds’, that is formed of a particle prefixed to a verb. The particle does not move along with the verb stem, but remains in clause-final position, even in main clauses:

- (22) Jan leest het boek uit  
 John reads the book PART  
 ‘John is finishing the book’

The separable compound verb *uitlezen* ‘to finish (a text)’ is formed of the verb *lezen* ‘to read’ plus the particle *uit*, literally ‘out’. In (22), this particle is stranded in sentence-final position, although it is part of the verb. We could speculate that this particle is a P-word, which means it can stand on its own. As a result, it is not necessary to move it along with the verbal stem to the V2 position.

Dutch has another type of compound verb, the so-called ‘inseparable compounds’. They also consist of a particle plus a verb stem, but contrary to the separable compounds, the particle cannot be separated from the verb. Therefore, when such a verb appears as finite verb in a main clause, the particle moves along with V:

- (23) Jan voorkomt erger  
 John prevents worse  
 ‘John prevents worse from following’

The verb *voorkomt* in (23) is formed of the verbal stem *komen* ‘to come’ plus the particle *voor* ‘for, before’. This particle cannot be separated from the verb stem. We could assume that such a particle does not form an independent P-word, and as a consequence cannot stand on its own. Therefore, when T is attracted by C and drags along V, the particle is dragged along with it.

The Arabic example in section 3.1 also provides an instance of head adjunction. The definite article *al-* is always prefixed to the noun. As we can see in the linear structure of (10b), repeated here as (24), the D is adjacent to Num, and Num is adjacent to N. As a result, the three heads can be morphologically fused, which in fact they are.

- (24) a. 'akaltu -l-ṭa<sup>c</sup>ām-a -l-ladīd-a  
 I ate the-food-ACC the-delicious-ACC

- ‘I ate the delicious food’
- b. D Num N D\*

It must be kept in mind that these remarks on head movement are very preliminary. But to my mind they offer a very attractive perspective. If the analysis here is in the right direction, it may become possible to link head movement to phonology. We can see how two heads can adjoin and form a single P-word, and we can furthermore see how a large chunk of lexical material moves when one head attracts only one other head.

If such a link between syntax on the one hand and morphology and/or phonology on the other could indeed be made in a principled way, it would be an important step forward in the field of linguistics. It would also provide syntacticians with an extra empirical test: heads that are morphologically fused, should end up adjacent in the linear structure. If they do not, there must something more must be at work. One influence disturbing the picture might be historical development.

## 4.2 An historical perspective

When we apply the ideas in the previous section to English, we stumble upon some problems. I will shortly discuss the matter here, and suggest a direction in which the solution may be found.

When we look at the linear structure of the Dutch (sub)clause, we see that the two fused heads V and T are not only adjacent, but they are also ordered in the correct way: the tense affix on the verb is a suffix, and likewise, T follows V:

- (25) C D<sub>su</sub> D<sub>ob</sub> V T

When we look at the English structure, however, we see that T and V are ordered in the wrong way:

- (26) D<sub>su</sub> T V D<sub>ob</sub>

At first, this seems to be a problem. But note that for compound tenses, the order is in fact correct:

- (27) John has read the book

The auxiliary, carrying the features of T, precedes the participle, which carries the features of V.<sup>16</sup> Furthermore, we know that the structure of modern English is in fact a rather recent development. Older forms of English showed a word order that is more consistent with that of other Germanic languages. The morphological fusion of V and T in English took place in that stage. When the structure of English later changed,<sup>17</sup> the language had to work with the fused V-T forms it possessed at that moment. The restructuring affected the word order, but it could not affect the fusion process, since that had been completed a long time before.

<sup>16</sup>In Dutch, the order of compound tenses is somewhat problematic. The expected order, participle - auxiliary is in fact grammatical, but contrary to expectation, the reverse order is, too. I have no explanation to offer for this. It is, however, a well-known and still unresolved problem of Dutch syntax. Note, by the way, that in German, only the expected order is grammatical.

<sup>17</sup>A change that can easily be described in the current framework. What is actually changing is the settings for the linearization parameters of the heads.

If these tentative remarks are in the right direction, diachronical evidence may be used to corroborate a synchronic analysis. If we had no historical data for English, we might be led to think that the linearization analysis of English is incorrect, since it puts T before V, whereas in the fused forms, T follows V.<sup>18</sup> But since we are aware of the historical development of English, we can actually explain the anomaly.

### 4.3 Linearization parameters

It could be argued that it is not very attractive to have two linearization parameters that can be set for each category independently. It would be much more preferable if there were some principled way to explain word order. I am not sure, however, if that is really preferable. After all, we cannot deny that language consists to a certain extent of arbitrary facts. The lexicon is the prime example. The fact that in the English-speaking part of the world a tree is called ‘tree’ is just an arbitrary fact of life. The word could well have been ‘beam’, like the Dutch and the German word for tree, *boom* and *Baum*, respectively.

The question to be asked then is what makes it possible for a certain aspect of language to be arbitrary. Or in other words: when is it important that something be done in a generalized, systematic way?

The answer to this question, I believe, is not so much a matter of philosophical necessity, but rather of practical necessity and of empirical fact. Although the English language arbitrarily chose to use ‘tree’ rather than ‘beam’ to name the property tree, the word must be used in a systematic way. Every speaker of the language must use this word in that meaning. If he does not, communication will suffer.

When we look at another property of language, for example that of meaning composition, we might argue that since humans form a single biological race, it is to be expected that meaning formation, which takes part in the brain, takes place in the same way in all humans, no matter what language they happen to speak. This is an empirical necessity for the simple fact that the structure and composition of the human brain is to a large extent identical in all humans.

It seems obvious that the matter of forming a linear structure out of a hierarchical one is closer to that of the lexicon than to that of meaning composition. It is not important how a tree structure is linearized, just as long as all speakers of a linguistic community do it in the same way.

This is not to say that linearization does not take place in the brain. Obviously, it does. What I am arguing is that the universal aspects of linearization are not on the surface, but one level deeper, in the same way that the universals of the lexicon are not at the surface. There are clear and universal principles governing the set of possible word forms. A language has, however, a certain freedom in applying these principles. In the same way, there are universal principles governing the linearization of hierarchical structures, but a certain variation is possible. If it turns out to be possible to describe this variation with only two parameters, this would in fact be a very attractive description.

---

<sup>18</sup>Of course, the data presented by the compound tenses actually supports the analysis.

## 5 Conclusion

All in all, I believe that the paper shows that the principle of recursive linearization is promising. The system is powerful enough to allow for a large variety in word order, yet it is precise enough to make some specific and empirically testable predictions. The combination of these two properties make the system worthy of further research.

## References

- Chomsky, N. (1995). *The Minimalist Program*, Cambridge, Mass: The MIT Press.
- Chomsky, N. (1998). Minimalist inquiries: the framework, MIT Occasional Papers in Linguistics 15.
- Chomsky, N. (1999). Derivation by phase, MIT Occasional Papers in Linguistics 18.
- Fassi Fehri, A. (1999). Arabic modifying adjectives and DP structures, *Studia Linguistica* **53**(2): 105–154.
- Kayne, R. (1994). *The antisymmetry of syntax*, Cambridge, Mass: The MIT Press.
- Kremers, J. (in preparation). The noun phrase in Arabic, University of Nijmegen.
- Ritter, E. (1991). Two functional categories in noun phrases: evidence from Modern Hebrew, in S. Rothstein (ed.), *Perspectives on Phrase Structure: Heads and Licensing*, Vol. 25 of *Syntax and Semantics*, San Diego: Academic Press, pp. 37–62.
- Shlonsky, U. (2000). The form of the Semitic noun phrase: an antisymmetric, non N-movement account, ms. University of Geneva.