University of Massachusetts Occasional Papers in Linguistics

Volume 17 *University of Massachusetts Occasional Papers -- Volume 15: Issues in Psycholinguistics*

Article 9

1991

Deterministic Parsing and the Verb Raising Construction in German and Dutch

Hotze Rullmann
University of Massachusetts at Amherst

DETERMINISTIC PARSING AND THE VERB RAISING CONSTRUCTION IN GERMAN AND DUTCH

Hotze Rullmann

Department of Linguistics
University of Massachusetts at Amherst

In this paper I will take up two questions pertaining to the processing of the well known Verb Raising construction in German and Dutch. The first question is: Is it possible to parse such constructions in a strictly deterministic fashion? The hypothesis that all natural languages are parsable strictly deterministically has been put forward by Marcus (1980). SOV languages like German and Dutch obviously are a challenge for this hypothesis, since many decisions the parser has to make early on depend on the properties of the verb, which in these languages appears at the end of the sentence. The second question I will discuss is: Why are Verb Raising constructions in German often more difficult to process than their counterparts in Dutch? (see Bach *et al.* 1986) The only relevant syntactic difference between the two languages is that in Dutch the order of the verbs in the sentence final cluster is the reverse of that in German.

The structure of the paper is as follows: Section 1 outlines Marcus's Determinism Hypothesis and gives some examples of artificial languages that do not appear to be parsable in a strictly deterministic fashion. In section 2, I argue that German is in fact very similar to one of these artificial languages and is therefore problematic for the Determinism

Hypothesis. In section 3 and 4, I show how the Verb Raising construction in German can be parsed strictly deterministically if we use a formalism for the representation of phrase structure proposed by Lasnik and Kupin (1977) and in unpublished work by Huybregts. In section 5, finally, the same formalism is used to state parsing rules for the Verb Raising construction in Dutch. The fact that these rules can be applied at an earlier stage in the parsing process than their German counterparts may explain why the Verb Raising construction is more difficult to process in German than in Dutch. In the Appendix some formal definitions are given which are referred to in the main text.

1. The Determinism Hypothesis

1.1 *Deterministic Parsing*

Marcus (1980) proposes the hypothesis that "the syntax of any natural language can be parsed by a mechanism which operates 'strictly deterministically' in that it does not simulate a nondeterministic machine".[1] He calls this the Determinism Hypothesis. A nondeterministic machine is a machine whose operations are underdetermined by its input and internal state. At each point in its course of actions it may choose one out of several options, having so to speak a magical oracle telling it what to do. Physically existing machines are of course deterministic, but they can *simulate* a nondeterministic machine either by pseudo-parallelism or by using backtracking. (A parallelistic machine explores all possible paths through the search space at the same time and its search is successful if one of those paths ends in a final state. A machine using backtracking, on the other hand, pursues just one path, but it can trace back its steps and undo a decision that has turned out to be wrong.) The Determinism Hypothesis, then, comes down to the claim that neither pseudo-parallelism nor backtracking is necessary for the parsing of natural languages.

To clarify what the implications of the Determinism Hypothesis are, let us consider the following pair of sentences:

---

[1] Marcus (1980), page 2.

(1) a. Is the block sitting in the box?
b. Is the block sitting in the box red?

In (1a) the phrase *sitting in the box* is the VP of the main sentence, but in (1b) it is a modifier of the subject NP. A parser scanning either sentence from left to right and building up syntactic structure as it goes along has to make a decision upon reaching the word *sitting*. It can either make *sitting* the main verb of the sentence, or it can attach it as a modifier to the subject NP. Which choice is the right one depends on the presence of the word *red* at the end of the sentence. A machine simulating nondeterminism can solve this problem in two ways. It can either build both structures at the same time and discard at the end of the parse whichever of the two analyses turns out to be wrong, or it can build only one of the two structures and then backtrack if necessary. In both cases syntactic structure that has been built up by the parser is destroyed later on.

A strictly deterministic parser cannot employ either of these two strategies. It simply has to make the right decision immediately. To ensure this, Marcus proposes that the following three restrictions be placed on a strictly deterministic parser:
- All syntactic substructures created by the parser are permanent. Nodes cannot be destroyed; features cannot be removed; the attachment of a daughter node to its mother cannot be broken.
- All syntactic substructures created by the parser for a given input must be output as part of the syntactic structure assigned to that input.
- No temporary syntactic structures are encoded within the internal state of the machine.

Given these constraints, a parser must be able to build up syntactic structure without ever making mistakes. In other words: every hypothesis the parser makes about the syntactic structure of its input string must turn out to be right in the end.

The question then arises, how a strictly deterministic parser can resolve local ambiguities of the sort exemplified in the sentences in (1). Marcus argues that a deterministic parser must have some kind of "look-ahead", *i.e.* a device which allows it to inspect more than one word in the input sentence at the same time. In the course of an analysis of (1b), for instance, such a look-ahead facility would enable the parser to "see" the word *red* at the moment at which the

relation of the verb *sitting* to the NP *the box* has to be determined.

It will be clear that the parser must not be able to look ahead over arbitrary distances. If the parser had an unlimited capability for looking ahead in the sentence, it would have little difficulty in making the right decisions all the time, and therefore the notion of deterministic parsing would lose its content. Marcus (1980) describes a parser for English with a look-ahead device, called the buffer, which is a "window" consisting of three cells. Each cell of the buffer can contain one word or constituent of the input sentence. As a consequence, the parser is able to look ahead over a distance of no more than three constituents of the input sentence. Thus, at most two constituents may intervene between the point at which a local ambiguity occurs and the item that serves as a clue for resolving it.

If we want to restrict the look-ahead capability of the parser, we should of course not only limit the size of its actual look-ahead "window", but also make sure that the parser does not have any other devices that could be used to enlarge its *de facto* look-ahead capability. Suppose for instance, that apart from a limited look-ahead window, the parser has a stack in which lexical items can be stored. In order to postpone an essential decision about syntactic structure, such a parser could just push all lexical items it encounters onto the stack until a clue resolving the ambiguity becomes visible. In effect such a parser would have an unlimited look-ahead.

1.2 *Two artificial languages not parsable by a deterministic parser*

We may ask what the class of languages is that can be parsed by a deterministic parser with a limited look-ahead window (and without any device that can be used to circumvent this restriction by acting as a pseudo look-ahead). In this section, I will show that there are certain artificial languages that cannot be handled by such a parser. This result will not depend on the actual size of the look-ahead, however, but only on the fact that it is finite. Not even a deterministic parser with a very large, but finite, look-ahead window would be able to deal with the languages described

below.

Consider the following context free rewrite grammar:

(2)
S -> A B	B -> b B
S -> C D	B -> e
A -> a	D -> b D
C -> a	D -> d

The language generated by this grammar is (in Kleene star notation) $ab^*(e,d)$. Parsing the language according to the above grammar is impossible for a deterministic parser with a finite look-ahead. The reason is that if the parser wants to decide whether the a at the beginning of the input string is dominated by A or C, it has to look ahead to the last word of the string. If this last word is an e, then the first word is a dominated by A; otherwise it is dominated by C. The number of words between the first word and the last word of the string is unlimited, however.

A second example of a language not parsable by a deterministic parser with a limited look-ahead is the one defined by the following grammar:

(3)
S -> S x
S -> a S y
S -> a a S z
S -> a v

A sentence generated by this language consists of a string of a's followed by a string of elements from the set {v,x,y,z}. Some examples of grammatical sentences are given in (4):

(4)
a. [a[av]y]
b. [aa[a[av]y]z]
c. [a[aa[av]z]y]

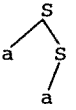
Consider (4b) and (c). Both begin with a string of four a's, but they should be parsed in different ways. In (4b), the first two a's are sisters and they are both directly dominated by the topmost S. In (4c), only the first a is a daughter of the topmost S, while the second and third a are daughters of a lower S node. Now imagine a parser parsing either (4b) or (4c). As a first step, the parser can create the topmost S node and attach the first a to it:

(5)

Next the parser must either attach the second a to the same S-node, or it must create a new (embedded) S-node and attach the second a to the latter. In the first case, the beginning of the sentence is analyzed as in (6a); in the second case, as in (6b):

(6) a.

b.



At the point at which the parser has to choose between (6a) and (6b), though, the parser does not have the information it needs. As (4b) and (c) show, the order of the y and z at the end of the sentence is crucial for deciding between the two analyses, but they are not yet visible to the parser. Enlarging the look-ahead window would help in this specific case, but not in general, because the y and z may be arbitrarily far away. We may conclude that the language generated by the grammar in (3) cannot be parsed by a parser which operates strictly deterministically and has a finite look-ahead.

2. Verb Raising: Evers's Analysis

2.1 *Embedded infinitival clauses in German*

In the preceding section, we have discussed two examples of artificial languages that cannot be parsed by a strictly deterministic parser. There are certain constructions in German and Dutch that seem to have very similar properties. Therefore these languages are problematic for the Determinism Hypothesis.

Dutch and German are SOV languages.[2] Verbs in the two languages can take different kinds of sentential complements. Finite complement clauses appear to the right of the matrix verb, whereas infinitival complement clauses appear either to the right or to the left of the verb. In what follows, I will only be concerned with infinitival complements occuring to the left of the verb. Infinitival complement clauses may or may not have an overt subject. Using traditional terminology, I will call verbs that take an infinitival complement with an overt subject, ACI verbs (*Accusativus Cum Infinitivo*). Among the verbs taking complements without an overt subject, we can distinguish between subject control, object control, and raising verbs. A special class of subject control verbs is that of the modal verbs.[3] Some matrix verbs require an infinitive marker (*te* in Dutch and *zu* in German) on their complement clause. Some of the verbs that take infinitival complements are listed in (7):

(7) a. Modal verbs:
Dutch: *willen, kunnen, mogen, moeten*
German: *wollen, können, dürfen, müssen*
('want', 'can', 'be allowed', 'must')

b. Subject control verbs:
Dutch: *proberen, beloven, vergeten, hopen, leren*
German: *versuchen, versprechen, vergessen, hoffen, lernen*
('try', 'promise', 'forget', 'hope', 'learn')

c. Object control verbs:
Dutch: *bevelen, vragen, leren*
German: *befehlen, bitten, lehren*
('order', 'ask', 'teach')

d. Raising verbs:
Dutch: *schijnen*
German: *scheinen*
('seem')

e. ACI Verbs:
Dutch: *zien, horen, laten*
German: *sehen, horen, lassen*
('see', 'hear', 'let')

---

[2] In main clauses the main verb is moved to the second position of the sentence, however. All the examples, therefore, involve embedded clauses.

[3] I assume that modal verbs take clausal complements, but this is not essential.

Apart from verbs taking sentential complements, both languages have auxiliary verbs (*zijn* and *hebben* in Dutch and *sein* and *haben* in German). I will assume that auxiliary verbs take VP-complements rather than S-complements, although nothing in what follows crucially depends on this assumption. The verb which is the head of such a VP-complement must be a past participle.[4] In (8) a number of German examples with various sorts of verbs are given:

(8) a. weil Klaus [PRO ein Nilpferd kaufen] will
because Klaus a hippo buy wants
'because Klaus wants to buy a hippo'

b. weil Klaus [ein Nilpferd gekauft] hat
because Klaus a hippo bought has
'Because Klaus has bought a hippo'

c. weil Klaus [PRO uns ein Nilpferd zu
because Klaus us a hippo to
verkaufen] versprach
sell promised
'because Klaus promised to sell us a hippo'

d. weil ich [PRO Klaus [PRO das Nilpferd
because I Klaus the hippo
füttern] helfen] muß
feed help must
'because I must help Klaus to feed the hippo'

e. weil ich [PRO [Klaus ein Nilpferd füttern]
because I Klaus a hippo feed
sehen] will
see want
'because I want to see Klaus feed a hippo'

Characteristic of such constructions is the cluster of verbs at the end of the sentence, which is preceded by a string of NP's. The sentences in (8) reflect the underlying order of the constituents. At the surface, some complement clauses may be extraposed, and certain reorderings of the verbs in the sentence-final cluster are also allowed (see den Besten and Edmondson (1983)). In some cases native speakers actually prefer extraposition or reordering of the

---

[4] If a verb that is the complement of an auxiliary, has an infinitival complement, however, it will - under certain circumstances - appear in the infinitival rather than in the participial form. This phenomenon, known as the Double Infinitive Construction, will not be discussed in this paper. See den Besten and Edmondson (1983).

verbal cluster to leaving the verbs in their underlying order. In this paper, however, I will disregard extraposition of clausal complements and moreover pretend that, in German, the only permissable order of the verbs in the sentence final cluster is the underlying order we find in (8).[5]

Now let us consider why sentences such as the ones in (8) are problematic for the Determinism Hypothesis. The string of NP's in these sentences is divided up by clause boundaries, and thus the NP's in the string may belong to different clauses. Importantly, it depends on the verbs in the cluster at the end of the sentence which NP belongs to which clause. This may give rise to local ambiguities of the sort we saw above for the language specified by the grammar in (3). Compare for instance the following two sentences:

(9) a. weil ich [Klaus das Nilpferd füttern] sah
because I Klaus the hippo feed saw
'because I saw Klaus feed the hippo'

b. weil ich Klaus [PRO das Nilpferd füttern]
because I Klaus the hippo feed
half
helped
'because I helped Klaus to feed the hippo'

In (9a) *Klaus* is the subject of the infinitival complement of the matrix verb, whereas in (9b) it is the object of the matrix verb. This local ambiguity is very similar to the one between (4b) and (c). A deterministic parser for German, will therefore have similar difficulties in coping with sentences of this kind. When *Klaus* has to be attached to its mother node, the verbal cluster may not be visible to the parser, since all kinds of lexical material (such as adverbs and adverbial PP's) may intervene, as in (10a) and (b):

---

[5] There is a lot of dialect variation as to the order of the verbs in the sentence final cluster. If den Besten and Edmondson (1983) are correct, West-Frisian and Low German are dialects which only permit the verbs to appear in the underlying order. The idealization made in this paper is that Standard German is such a dialect, too.

(10) a. weil ich Klaus gestern nach acht
because I Klaus yesterday after eight
uhr im Tiergarten das Nilpferd
o'clock in-the Zoo the hippo
füttern sah
feed saw
'because I saw Klaus feed the hippo in the Zoo yesterday after eight o'clock'

b. weil ich Klaus gestern nach acht
because I Klaus yesterday after eight
uhr im Tiergarten das Nilpferd
o'clock in-the Zoo the hippo
füttern half
feed helped
'because I helped Klaus to feed the hippo in the Zoo yesterday after eight o'clock'

Other examples of such local ambiguities that can be resolved only by inspecting the sentence final verb cluster, are given in (11) and (12):

(11) a. weil Klaus uns [PRO sein Auto zu
because Klaus us his car to
verkaufen] bittet
sell asks
'because Klaus is asking to sell his car'

b. weil Klaus [PRO uns sein Auto verkaufen]
because Klaus us his car sell
wants
will
'because Klaus wants to sell us his car'

(12) a. weil ich [PRO Klaus [PRO das Nilpferd
because I Klaus the hippo
füttern] helfen] muß
feed help must
'because I must help Klaus to feed the hippo'

b. weil ich Klaus [PRO das Nilpferd zu
because I Klaus the hippo to feed
füttern] versprach
feed promised
'because I promised Klaus to feed the hippo'

It is clear, then, that constructions of this kind pose a serious problem for strictly deterministic parsers with a limited look-ahead capability, and hence for the Determinism Hypothesis.

2.2 *Evers's analysis of Verb Raising constructions*

We should not jump to conclusions too hastily, however. Sentences with verb clusters consisting of three verbs or more, like (8d) and (e), are reported not to be very easily processable even by native speakers. When we add more embeddings, the sentences become still less acceptable (see Bach *et al.* (1986)). This seems to indicate that the human sentence processor does have some difficulties in handling the verbal complex in German. Note however that the problem for the determinism hypothesis already arises with sentences like (9) - (11), which have verb clusters consisting of just two verbs. Native speakers do not have any difficulties in processing these sentences, even if a lot of lexical material precedes the verb cluster, as in (10). Therefore, it remains to be explained how such sentences can be parsed deterministically. In section 5, I will come back to the observations made by Bach *et al.* (1986) concerning the processability of sentences with complex verb clusters.

So far I have tacitly assumed that the (surface) structure of the sentences under consideration is essentially as shown in (8) - (12). Evers (1975), however, who gives the classic analysis of such constructions, argues that these structures are the deep- but not the surface-structures. He proposes a rule, called *Verb Raising*, which adjoins the verb of the embedded clause to the matrix verb. This can be seen most clearly in Dutch, because in this language the surface order of the verbs in the cluster is the reverse of the German order.[6] So, for instance, the Dutch equivalences of the sentences in (8) are:

[6] Again this is an idealization of the data. Alternative orders of the verbs in the sentence final cluster are sometimes allowed in Dutch. See den Besten and Edmondson (1983).

(13) a. omdat Klaus een nijlpaard wil kopen
because Klaus a hippo buy wants
'because Klaus wants to buy a hippo'

b. omdat Klaus een nijlpaard heeft gekocht
because Klaus a hippo has bought
'because Klaus has bought a hippo'

c. omdat Klaus ons een nijlpaard beloofde te
because Klaus us a hippo promised to
verkopen
sell
'because Klaus promised to sell us a hippo'

d. omdat ik Klaus het nijlpaard moet helpen
because I Klaus the hippo must help
voeren
feed
'because I must help Klaus to feed the hippo'

e. omdat ik Klaus een nijlpaard wil zien
because I Klaus a hippo want see
voeren
feed
'because I want to see Klaus feed a hippo'

According to Evers, these sentences are derived from deep structures parallel to the structures in (8) by Verb Raising. In Dutch, this rule adjoins the verb of the embedded clause to the right of the matrix verb, as shown for (13a) in (14):

(14) omdat Klaus [PRO een nijlpaard *v*] [wil kopen]

(*v* denotes the trace of the verb.)
Because Verb Raising applies cyclically, more complex examples like (13d) and (e) can also be derived.

Interestingly, Evers shows that Verb Raisings also applies in German, the only difference being that in this language the raised verb is adjoined to the *left* of the matrix verb, rather than to the right. As a consequence, Verb Raising does not affect the order of verbs in German. There is positive evidence, showing that Verb Raising also applies in German. Evers shows that Verb Raising has a number of syntactic effects, which he adduces to the pruning of the S and VP-node dominating the deep structure position of the raised verb. One of these effects concerns the scope of sentential adverbs. In both (15a) and (15b) the negating adverb (*niet* and *nicht*, respectively) can have scope over the entire clause:

(15) a. omdat hij de hond niet wil verkopen
because he the dog not wants sell
b. weil er den Hund nicht verkaufen will
because he the dog not sell wants
'because he does not want to sell the dog'

Since, in general, negating adverbs can only have scope over the minimal clause containing them, these data show that in German, as well as in Dutch, the embedded clause somehow is made transparent for scope. Although it is not clear whether, as Evers claims, pruning of the S-node is involved, it is plausible to assume that this transparency is due to the raising of the embedded verb. Thus, these data (and other evidence I will not discuss here) support Evers's claim that Verb Raising also applies in German.

According to Evers, Verb Raising results in an essentially flat structure after pruning of the S and VP-nodes. The surface structure of (9a), for instance, whose deep structure is given in (16a), will be (16b):

(16) a. weil [$_S$ ich [$_{VP}$ [$_S$ Klaus [$_{VP}$ das Nilpferd füttern]] sah]]
b. weil [$_S$ ich [$_{VP}$ Klaus das Nilpferd [$_V$ füttern sah]]]

In (16b) the NP's *Klaus* and *das Nilpferd* are sisters, whereas in (16a) they are not. In general, Verb Raising constructions, in Evers's analysis, have the following form at the surface:

(17) [$_S$ NP1 [$_{VP}$ NP2 NP3 .... NPn [$_V$ ....]]]

Now notice that such flat structures would be easy to parse deterministically, in contrast to non-flat structures like (16a). The only thing the parser would have to do is to attach NP1 to the top S-node, then create a VP-node and attach all the following NP's to it. Local ambiguities of the sort we have seen above do not arise, because for instance (9a) and (9b) have isomorphic surface structures. The surface structure of (9a) is given in (16b) and that of (9b) in (18):

(18) weil [$_S$ ich [$_{VP}$ Klaus das Nilpferd [$_V$ füttern half]]]

If we want a parser to produce flat structures of this kind, the determinism problem simply disappears. The question, then, is whether we really want such

structures as output.

One objection against flat structures like (16b) and (18), is that the rule of pruning, which is necessary to derive them, has become disreputable in modern syntactic theory. Extensive pruning of nodes is not a possible operation in current theories of syntax. Another reason why we may not want a parser to output flat structures is that thematic relations cannot be directly recovered from such structures. Thematic relations are important, because, first, we must check whether there is a one-to-one relation between thematic roles and argument NP's, for instance in order to rule out as ungrammatical sentences with a transitive verb which don't have an object; and second, because recovering thematic relations is a necessary precondition for doing semantics. Pruning S and VP-nodes involves the annihilation of syntactic information that is useful for determining thematic relations. It may be possible to find a way of recovering the thematic relations in a flat structure in some non-syntactic way, for instance by means of a mechanism for the composition of theta-structures. This would mean that we shift part the burden of interpreting the sentence from syntax to some other component of the grammar, thereby making the task of the parser less difficult. Although this would be a legitimate move, I will pursue another course in this paper in that I want the parser to recover as much syntactic information as possible.

In the next section, I will discuss a recent approach to Verb Raising, which differs from that of Evers in that it does not assign flat surface structures to Verb Raising constructions. After that, in section 4, the consequences of this approach for deterministic parsing will be discussed.

3. Verb Raising: Huybregts's Analysis

3.1 *Verb Raising as reanalysis*

An alternative to Evers's analysis of Verb Raising has been developed by Huybregts. His work on this phenomenon has never been published, but Haegeman and Van Riemsdijk (1986) give a short impression of Huybregts's approach and develop it further in order to account for what they call 'Verb Projection Raising', a variant of Verb Raising found in West-Flemish and

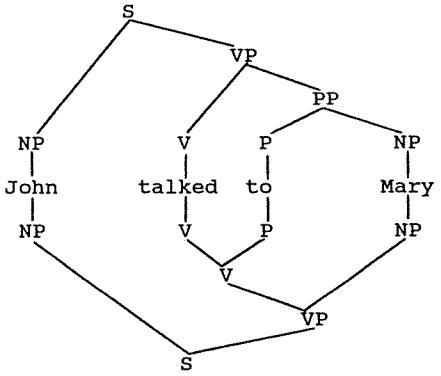
Zurich German. In this paper, I will not consider Verb Projection Raising, but I will make use of Haegeman and Van Riemsdijk's version of Huybregts's theory. All references to Huybregts which appear below should be understood as referring to Huybregts's account as it is reported by Haegeman and Van Riemsdijk.

Huybregts treats Verb Raising not as a verb movement rule, but as a kind of reanalysis. A paradigmatic example of reanalysis is that of a verb and a preposition in pseudo-passives, as in the following example:

(19) Mary was talked to

It is argued that the verb *talked* and the preposition *to* have to be (re)analyzed as one verb in order for pseudo-passivization to be possible. In Huybregts's theory, reanalysis results in a syntactic object that is not representable as a single tree, but only as a set of trees. After reanalysis a sentence like *John talked to Mary* would be associated with two trees:

(20)



What the double tree representation in (20) expresses, is that the phrase *to Mary* is a PP, while at the same time *talked to* is a V. Note that this information cannot be represented in a single tree.

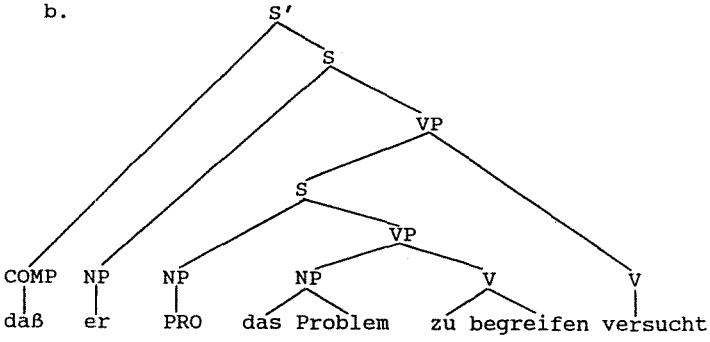
In general, phrase structure trees can be regarded as representing sets of *is a* statements. Thus, the upper tree of (20) represents the following statements: *John talked to Mary* is an S; *talked to Mary* is a VP; *to Mary*

is a PP; *John* is an NP; *talked* is a V; *to* is a P; and *Mary* is an NP. This set of statements is *consistent* in the sense that it is representable as a single phrase structure tree. But if we add the statement "*talked to* is a V" to this set, it is no longer consistent. In Huybregts's analysis, then, the output of reanalysis is an inconsistent set of *is a* statements, which cannot be represented as a single phrase structure tree.

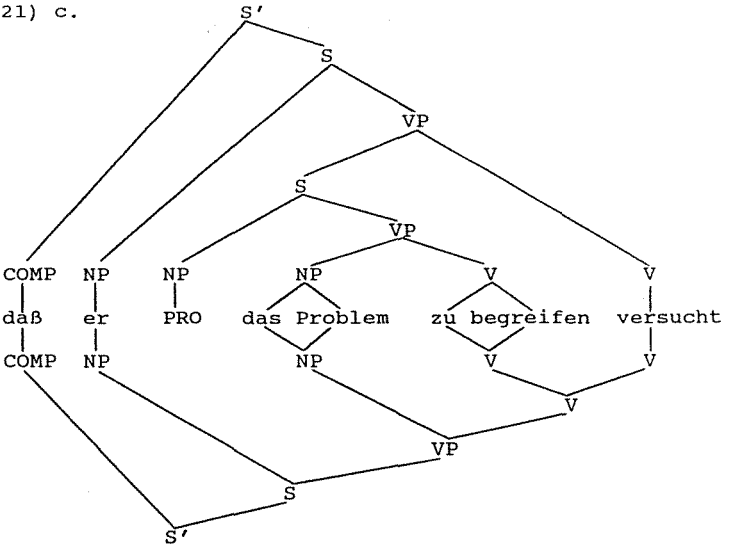
With respect to Verb Raising, Huybregts assumes that in the syntax only reanalysis takes place, and that the inversion of verbs we see in Dutch applies in the phonological module of the grammar (after reanalysis). For the moment, we will concentrate on Verb Raising in German which does not have inversion. Huybregts's rule of Verb Raising reanalyzes the main verb of an embedded clause and the matrix verb governing this clause as a single (complex) verb. A sentence like (21a), for instance, which before Verb Raising is associated with the one-dimensional representation (21b), will be associated with the two-dimensional representation (21c) after Verb Raising:

(21) a. daß er das Problem zu begreifen versucht
that he the problem to understand tries
'that he tries to understand the problem'

b.



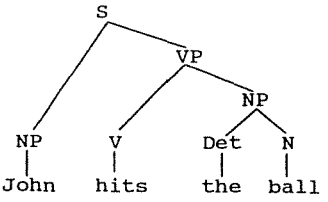
(21) c.



3.2 *Reduced Phrase Markers*

The Verb Raising rule which changes (21b) into (21c) can be formulated as a rule that adds an *is a* statement to the set of *is a* statements corresponding to (21b). Huybregts uses the formalism proposed by Lasnik and Kupin (1977) to implement this idea. Lasnik and Kupin introduce the notion of *Reduced Phrase Marker* (RPM) as a formalization of syntactic trees. An RPM is a set of so-called *monostrings*, where each monostring can be regarded as representing an *is a* statement. Each monostring is an element from the set $W^*\epsilon W^*$, where W is the set of words (terminal elements) and ϵ is the set of category symbols (non-terminal elements). How monostrings are associated with *is a* statements, is explained most easily by giving an example. Consider the syntactic tree given in (22).

(22)



The RPM corresponding to this tree is the following set of monostrings:

(23) {*John hits the ball*,
NP *hits the ball*,
John V *the ball*,
John hits Det *ball*,
John hits the N,
John hits NP,
John VP,
S}

The monostring *John hits* NP from this RPM corresponds to the statement "*the ball* is an NP". In the same way, *John* VP corresponds to "*hits the ball* is a VP". In general, if w, v and u are strings of words (i.e. $w, v, u \circ W^*$), X is a category symbol (i.e. $X \circ \epsilon$), and γ is a RPM such that wXv, $wuv \circ \gamma$, then wXv corresponds to the statement that u is an X.

Lasnik and Kupin formulate conditions on the wellformedness of RPM's, which jointly guarantee that every RPM can be depicted as a single normal phrase structure tree. Huybregts's analysis, in contrast, admits RPM's which are not representable as single phrase structure trees, and so he has to weaken Lasnik and Kupin's wellformedness conditions. In Huybregts's theory, RPM's are allowed that can only be represented as multi-dimensional trees of the sort we saw above. In the Appendix, Lasnik and Kupin's wellformedness conditions are stated and it is shown how they can be modified so as to deal with RPM's of the sort proposed by Huybregts.

In (24) the RPM is given that corresponds to the phrase structure tree (21b):

(24) {*daß er das Problem zu begreifen versucht,*
COMP *er das Problem zu begreifen versucht,*
daß NP *das Problem zu begreifen versucht,*
daß er NP *das Problem zu begreifen versucht,*
daß er Det *Problem zu begreifen versucht,*
daß er das N *zu begreifen versucht,*
daß er das Problem V *versucht,*
daß er das Problem zu begreifen V,
daß er NP *zu begreifen versucht,*
daß er VP *versucht,*
daß er S *versucht,*
daß er VP,
daß S,
S'}

(24) is a consistent RPM: it can be represented as a single phrase structure tree. Note, by the way, how PRO is represented in this RPM. The monostring *daß er* NP *das Problem zu begreifen versucht* corresponds to the statement that between *er* and *das* there is an NP which dominates no lexical material, *i.e.* an empty NP.

What Verb Raising does in Huybregts's analysis, is to add a monostring to (24), so that it is no longer consistent. This added monostring corresponds to the statement that the string *zu begreifen versucht* is a V and it is given in (25):

(25) *daß er das Problem* V

The resulting RPM is only representable as a set of phrase structure trees as in (21c). A reanalysis rule, then, can be formulated as a rule for adding monostrings to RPM's. In (26) a rough formulation of Verb Raising, adapted from Haegeman and Van Riemsdijk (1986), is given:

(26) Let $w, v \in W^*$ and $x, y \in W$, such that x is a verb and y is a Verb Raising verb.
If Γ is an RPM such that $wxyv \in \Gamma$, then add wVv to Γ.

In the next section we will consider the way in which Huybregts's analysis can be put to use for the purpose of deterministic parsing.

4. Huybregt's Analysis and Deterministic Parsing

In section 2, we have seen that the flat surface structures proposed by Evers for Verb Raising constructions can be parsed strictly deterministically without special difficulties. In this section, I want to show that it is possible to formulate parsing rules producing non-flat, multi-dimensional structures of the sort proposed by Huybregts, which are strictly deterministic in a sense explained in section 4.3. These rules are stated in terms of a formalism for representing syntactic structure that is an adaptation of that of Lasnik and Kupin. After showing in section 4 how this works for German, I will show in section 5 that a modification of these rules suffices for Dutch Verb Raising constructions.

4.1 *Parsing with RPM's*

The Verb Raising rule we have formulated above (see (26)) is a rule that adds monostrings to RPM's. In terms of multi-dimensional trees like (21c), each application of Verb Raising adds another dimension to the tree. The idea underlying the parsing rules proposed in this section is that this process can be reversed. The parser first produces the bottom tree of (21c), which is essentially the flat surface structure of Evers's analysis, and then, by some special parsing rule, the other dimension is added. Or, in terms of the RPM formalism, the parser first produces a consistent RPM and then adds new monostrings to the set, which make it inconsistent.

In what follows, I will use a formalism for representing phrase structure which is different from, but closely related to, that of Lasnik and Kupin (1977). In (23), for instance, the monostring *John hits* NP can be said to correspond to the statement that *the ball* is an NP. In order to bring out the correspondence between monostrings and *is a* statements more clearly, I will use a pair notation. Monostrings, in my formalism, are replaced by ordered pairs <X, *w*>, where X is a category symbol ($X \in \Sigma$) and *w* is a (non-empty) string of words ($w \in W^*-\{e\}$). Every pair <X, *w*> now simply corresponds to the statement that the string *w* is a

constituent of the category X.[7] In my formalism the RPM associated with the sentence *John hit the ball* will be the following set of pairs:

(27) {<NP, *John*>, <V, *hit*>, <Det, *the*>, <N, *ball*>, <NP, *the ball*>, <VP, *hit the ball*>, <S, *John hit the ball*>}

In the Appendix, I give a formal definition of wellformedness for such RPM's.

Let us now discuss in some detail the parsing process that results in the RPM corresponding to the two-dimensional tree (21c). As a first step, the parser produces the RPM corresponding to the bottom tree of (21c). This RPM is given in (28) (in my notation):

(28) {<COMP, *daß*>, <NP, *er*>, <Det, *das*>, <N, *Problem*>, <V, *zu begreifen*>, <V, *versucht*>, <V, *zu begreifen versucht*>, <NP, *das Problem*>, <VP, *das Problem zu begreifen versucht*>, <S, *er das Problem zu begreifen versucht*>, <S', *daß er das Problem zu begreifen versucht*>}

As we have seen in section 2.2, this RPM, which corresponds to the flat structure of Evers's analysis, can be produced in a strictly deterministic way.

In the next step, the parser adds a few pairs to (28), so that an RPM results which corresponds to the multi-dimensional tree of (21c). The pairs that have to be added are the following:

(29) <VP, *das Problem zu begreifen*>
<S, *das Problem zu begreifen*>

We can formulate a rule that adds the two pairs in (29) to the RPM (28). A preliminary version of this rule is given in (30):

(30) Let v, $w \in W$ such that v is an infinitive (with marker *zu*) and w is a form of the verb *versuchen*.
Let $u \in W^*$.
If Γ is an RPM such that <V, v>, <V, w>, and <VP, uvw> $\in \Gamma$, then $\Gamma' = \Gamma \cup$ {<VP, uv>, <S, uv>}

---

[7] To make this really work we have to distinguish formally between different occurrences (tokens) of the same word (as in the sentence *The boy hit the ball*, where the word *the* occurs twice). We can do this by adding a unique index to every word (token) in the sentence.

If (30) is applied to (28), *u* is *das Problem*, *v* is *zu begreifen* and *w* is *versucht*.

Rules in the same format as (30) can be formulated for different classes of Verb Raising verbs. In (31), a classification of verbs is given that will be used in formulating these rules:

(31) AUX is the set of auxiliary verbs
MOD is the set of modal verbs
CON1 is the set of subject control verbs which don't have an object NP
CON2 is the set of subject control verbs which have an object NP
CON3 is the set of object control verbs
ACI is the set of verbs that take ACI complements
RAIS is the set of raising verbs
AUX, MOD, CON1, CON2, CON3, ACI, RAIS $\subseteq$ W

Along with (31), we need to distinguish between the various possible *forms* in which verbs can occur. To this end the following sets are defined:

(32) INF is the set of infinitives[8]
PART is the set of participles
INF, PART $\subseteq$ W

Now we can formulate the parsing rules. For each set in (31) there is a different rule. First, the rules are given in (33) without further comment, and after that, in section 4.2, the way in which they are applied is illustrated with an example.

[8] I will not distinguish between infinitives with and without the infinitive marker *zu*. All the examples will only involve verbs that take complements without *zu*. The rules could easily be modified so as to deal with infinitives that do have *zu*.

(33) Let $u,s,t \in W^*$ and $v, w \in W$. Let Γ be an RPM such that <VP, uvw> $\in \Gamma$.
i. If $w \in$ AUX and $v \in$ PART,
then $\Gamma' = \Gamma \cup$ {<VP, uv>}
ii. If $w \in$ MOD and $v \in$ INF,
then $\Gamma' = \Gamma \cup$ {<VP, uv>, <S, uv>}
iii. If $w \in$ CON1 and $v \in$ INF,
then $\Gamma' = \Gamma \cup$ {<VP, uv>, <S, uv>}
iv. If $w \in$ CON2, $v \in$ INF, $u = st$ and <NP, s> $\in \Gamma$,
then $\Gamma' = \Gamma \cup$ {<VP, tv>, <S, tv>}
v. If $w \in$ CON3, $v \in$ INF, $u = st$ and <NP, s> $\in \Gamma$,
then $\Gamma' = \Gamma \cup$ {<VP, tv>, <S, tv>}
vi. If $w \in$ ACI, $v \in$ INF, $u = st$ and <NP, s> $\in \Gamma$,
then $\Gamma' = \Gamma \cup$ {<VP, tv>, <S, stv>}
vii. If $w \in$ RAIS and $v \in$ INF,
then $\Gamma' = \Gamma \cup$ {<VP, uv>, <S, uv>}

4.2 *An example*

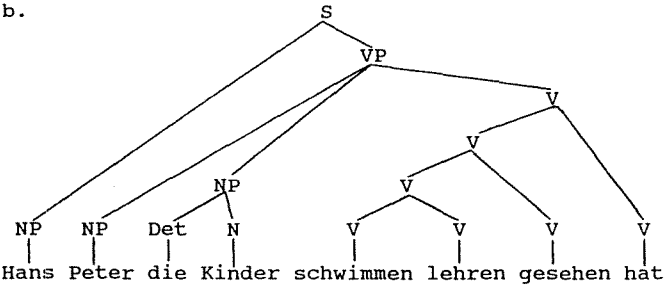
By way of example, I will demonstrate how the rules in (31) are applied in the parsing of a rather complex Verb Raising construction. Consider the following sentence:

(34) daß Hans Peter die Kinder schwimmen lehren
that Hans Peter the children swim teach
gesehen hat
seen has
'that Hans has seen Peter teach the children how to swim'

The parser first produces the RPM Γ in (35a), which corresponds to the flat tree in (35b):

(35) a.
Γ = {<NP, *Hans*>, <NP, *Peter*>, <Det, *die*>, <N, *Kinder*>,
<NP, *die Kinder*>, <V, *schwimmen*>, <V, *lehren*>,
<V, *gesehen*>, <V, *hat*>, <V, *schwimmen lehren*>,
<V, *schwimmen lehren gesehen*>,
<V, *schwimmen lehren gesehen hat*>,
<VP, *Peter die Kinder schwimmen lehren gesehen hat*>,
<S, *Hans Peter die Kinder schwimmen lehren gesehen hat*>}

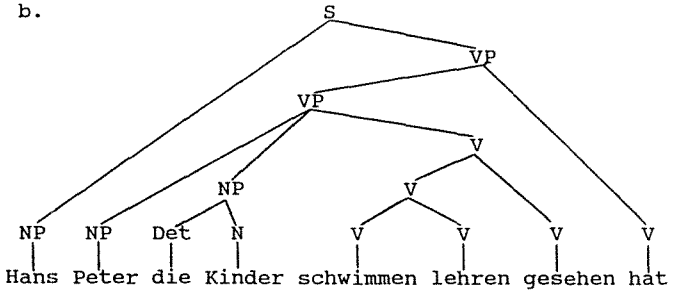
(35) b.



Rules from (33) are successively applied to γ. The first rule that can be applied is (i). *hat* is an auxiliary verb, *gesehen* is a participle and *Peter die Kinder schwimmen lehren gesehen hat* is a VP. Thus, if we apply rule (i), the variable *u* is instantiated as *Peter die Kinder schwimmen lehren*; *v* is instantiated as *gesehen*; and *w* is instantiated as *hat*. As a result, the pair <VP, *Peter die Kinder schwimmen lehren gesehen*> is added to γ. The resulting RPM, γ', is as stated in (36a). The tree depicted in (36b) is added as a second dimension to the representation of the sentence. (Note that (35b) and (36b) *together* correspond to γ'; these two trees are the two dimensions of the multi-dimensional tree associated with γ'.)

(36) a. $\gamma' = \gamma \cup$ {<VP, *Peter die Kinder schwimmen lehren gesehen*>}

b.

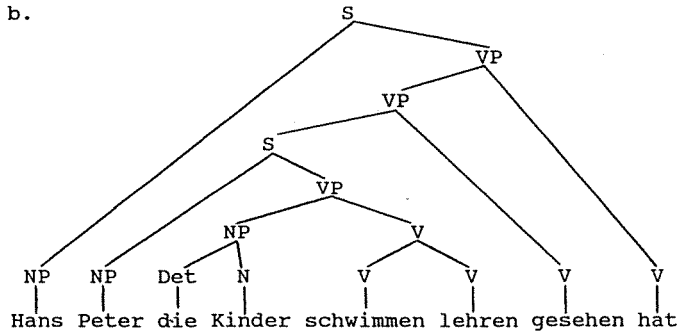


The next rule to be applied is (vi). This time *s* = *Peter*; *t* = *die Kinder schwimmen*; *v* = *lehren*; and *w* = *gesehen*. Note that (vi) can only be applied *after* (i), since the existence of the pair <VP, *Peter die Kinder schwimmen lehren*

gesehen>, which has been added to γ by the application of (i), is a necessary precondition for the application of (vi). The RPM resulting from the application of (vi) is γ'' as in (37a); the dimension added to the tree is shown in (37b).

(37) a. $\gamma'' = \gamma' \cup$ {<VP, *die Kinder schwimmen lehren*>, <S, *Peter die Kinder schwimmen lehren*>}

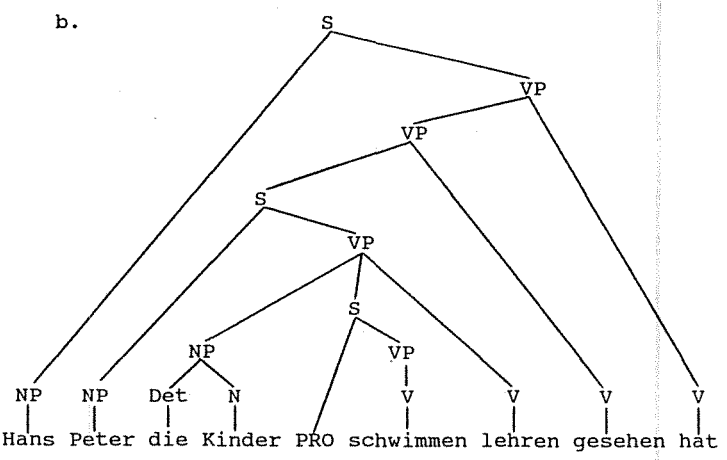
b.



Finally, (v) can be applied. This time *s* = *die Kinder*; *t* = *e* (the empty string); *v* = *schwimmen*; and *w* = *lehren*. The resulting RPM, γ''', is given in (38a), and the added dimension of the corresponding tree is shown in (38b):

(38) a. $\gamma''' = \gamma'' \cup \{\langle VP, \textit{schwimmen}\rangle, \langle S, \textit{schwimmen}\rangle\}$

b.



4.3 *Conclusion*

The question we started out with was whether the Verb Raising construction in German can be parsed strictly deterministically. Are the parsing rules proposed above strictly deterministic? The answer, of course, depends on how we define that term. Marcus's criterion for determinism is the following: a parser is strictly deterministic if all syntactic substructures it creates are permanent and cannot be discarded later on. In the standard phrase structure tree formalism this means that nodes cannot be destroyed, features cannot be removed and the attachment of a daughter node to its mother cannot be broken. In the RPM formalism in which the rules in (33) are stated the criterion will necessarily have a somewhat different interpretation, since the notions 'node', 'mother' and 'daughter' play no (direct) role in this formalism. The most straightforward idea seems to be the following: all the parser is allowed to do is to add new elements (*is a* pairs) to the RPM. No elements can be removed from the RPM or replaced by others. In that sense the rules in (33) are strictly deterministic.

It is important to realize, however, that adopting the RPM formalism and interpreting the notion of determinism in the way I just proposed, diminishes the empirical content of the Determinism Hypothesis quite a bit. The reason is that in an RPM certain syntactic information is not directly represented that is expressed in a standard phrase structure tree. What is lacking in the RPM is information about the relation of direct dominance, *i.e.* the mother-daughter relation. Consider for instance the RPM (35a) and its corresponding phrase structure tree (35b). In (35b) we see that the NP *Peter* is a daughter of the matrix VP. From the RPM (35a) this information can be recovered only in an indirect fashion, since the dominance and direct dominance are not primitive notions of the RPM formalism. In the Appendix, the notion *dominance*, which plays a role in the wellformedness conditions on RPM's, is defined as follows:

If A, B ϵ Γ, then A **dominates** B in Γ iff A =/= B and (for some X, Y ϵ Σ, x, y, w ϵ W^*) A = <X, xwy> and B = <Y, w>.

That is, A dominates B iff all the lexical material contained in A is also contained in B. *Direct dominance* can now be defined in terms of dominance:

If A, B ϵ Γ, then A **directly dominates** B in Γ iff A dominates B, and there is no C ϵ Γ such that C dominates B, B does not dominate C, A dominates C and C does not dominate A.[9]

In terms of these definitions, the pair <VP, *Peter die Kinder schwimmen lehren gesehen hat*> directly dominates <NP, *Peter*> in (35a). In (36a) we add the pair <VP, *Peter die Kinder schwimmen lehren gesehen*> to the RPM. Since this pair dominates the pair <NP, *Peter*>, the latter is no longer directly dominated by <VP, *Peter die Kinder schwimmen lehren gesehen hat*>. Thus, by adding new elements to the RPM, we can change relations of direct dominance. In contrast, changing the relations of direct dominance in

[9] Note that is not sufficient to require that there is no C such that C dominates B and A dominates C, because B and C may dominate each other, namely when B = <X, w> and C = <Y, w> (see Appendix). In that case it should be possible for A to directly dominate both B and C. A case in point is (46b) in which the pair <S, *the boy walks*> directly dominates both <VP, *walks*> and <V, *walks*>.

a phrase structure tree involves breaking mother-daughter attachments, something which is ruled out by Marcus's criterion of strict determinism.

In sum, an RPM crucially contains less information than the ordinary phrase structure tree it corresponds to. This makes it possible for a parser operating on RPM's to avoid making decisions about direct dominance, something which a parser operating on phrase structure trees cannot do. As a result, an RPM-based parser can parse certain constructions in a strictly deterministic fashion, which a tree-based parser cannot. The conclusion we can draw from this is that the empirical consequences of the Determinism Hypothesis depend on the specific formalism we adopt. What can be done strictly deterministically in one formalism may not be possible in an other formalism.

5. The Case of Dutch

5.1 *Bach, Brown and Marslen-Wilson (1986)*

In Dutch, the order of verbs in the verbal cluster is the inverse of the order in German. Thus, corresponding to the German example (34), repeated here as (39a), we have its Dutch counterpart (39b):

(39) a. daß Hans Peter die Kinder schwimmen lehren gesehen hat

b. dat Hans Peter de kinderen heeft zien leren zwemmen
that Hans Peter the children has seen teach swim
'that Hans has seen Peter teach the children how to swim'

Although the Dutch order is at first sight less "logical" because it has crossing rather than nesting dependencies, Bach *et al.* (1986) have shown that it is considerably easier to process. On the basis of comparative psycholinguistic experiments, they conclude that native speakers of Dutch have less difficulty in comprehending Verb Raising sentences in their own language than native speakers of German. Bach *et al.* suggest a very plausible explanation for this contrast. In this section, I will try to develop their suggestion in terms of the rules for parsing Verb Raising constructions proposed in the preceding section.

The suggestion made by Bach *et al.* is based on the observation that in a Dutch Verb Raising sentence like (39b) less deeply embedded verbs come before more deeply embedded verbs. In its German counterpart (39a) this is the other way around. For a sentence processor that analyzes the sentence from left to right this implies that, in Dutch, matrix verbs are available before embedded verbs. In (39b), for instance, the matrix verb *heeft* enters the parser before *zien*, which is the head of the embedded clause governed by *heeft*. *zien*, in its turn, precedes *leren*, the head of the clause governed by *zien*.

Bach *et al.* suppose that listeners compute partial interpretations of the sentence on-line. If this is correct, the fact that matrix verbs precede embedded verbs makes it possible to build a partial representation of the meaning of the sentence before the end of the sentence has been reached. In (39b), for instance, the processor can build the matrix *Hans heeft (VP)* as soon as the verb *heeft* is reached. In the corresponding German example (39a), the matrix cannot be built until the end of the sentence is reached. In German, embedded clauses can be built before their matrix clauses. A parser analyzing (39) can build the embedded clause *PRO schwimmen* as soon as it has reached the verb *schwimmen*, and only after that can it build the matrix clause *Peter die Kinder* (S) *helfen*.

Bach *et al.* conclude that "the most important variable in successful parsing and interpretation is not simply *when* information becomes available, but also *what* you can do with that information when you get it." The German listener can begin to build up embedded clauses, but "he has no higher structure into which to integrate this information at the time he receives it." The Dutch listener, on the other hand, starts out with matrix clauses into which the embedded clauses can be integrated later on. It appears, then, that it is easier to store incomplete structures whose higher level function is known than it is to store complete structures that are not yet integrated into some higher structure. The well known fact that center embedded structures (in English) are very hard to process points in the same direction.

The conclusions reached by Bach *et al.* are rather tentative and are stated very carefully. It is possible to find some support for their arguments, however, if we regard this issue from the point of view of the

parsing rules proposed in the preceding section. Consider for instance rule (33ii), repeated here as (40):

(40) Let $u \in W^*$ and v, $w \in W$. Let Γ be an RPM such that $\langle VP, uvw \rangle \in \Gamma$.
If $w \in$ MOD and $v \in$ INF, then $\Gamma' = \Gamma \cup \{\langle VP, uv \rangle, \langle S, uv \rangle\}$

For (40) to be applied, three conditions have to be satisfied:
- the sentence must contain a VP;
- the last word of this VP must be a modal verb;
- the word immediately preceding this modal verb must be an infinitive.

(40) is formulated for German. In a version for Dutch the last two conditions would have to be changed as follows:
- the *first* word of the verbal cluster in this VP must be a modal verb;
- the word immediately *following* this modal verb must be an infinitive.

Now consider the following two examples, in German and Dutch respectively:

(41) a. daß Peter Karl das Pferd springen lehren sehen will
that Peter Karl the horse jump teach see wants

b. dat Peter Karl het paard wil zien leren springen
that Peter Karl the horse wants see teach jump
'that Peter wants to see Karl teach the horse how to jump'

A parser for German analyzing (41a) from left to right has to check the three conditions stated above in order to apply (41). This implies that this rule cannot be applied before the parser has seen both *sehen* and *will*, which appear at the end of the sentence. In the Dutch sentence (41b), on the other hand, the conditions can be checked as soon as the first two verbs of the verbal cluster have been seen. Note that the parser does already know that there is a VP at that point. (In fact, the parser can expect a VP as soon as *Peter* has been analyzed as the subject of the sentence.) We may conclude that the Dutch equivalent of (40) can be applied earlier than its German counterpart. This explains why (41a) is easier to process than (41b), on

the plausible assumption that the possibility of applying rules like (40) "on the fly" during processing facilitates comprehension.

The question now is whether we can state formally the equivalent of rules like (40) for Dutch, and whether it is possible to apply these rules "on the fly", before the end of the sentence has been reached. To answer the first question, we must develop a formalism for describing discontinuous constituency (section 5.2); to answer the second question, we must find a way of dealing with *is a* statements about unknown lexical material (section 5.3).

5.2 *Discontinuous constituency*

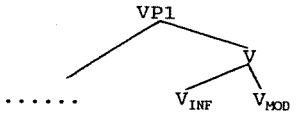
Consider again rule (40), repeated here as (42):

(42) Let $u \in W^*$ and v, $w \in W$. Let Γ be an RPM such that $\langle VP, uvw \rangle \in \Gamma$.
If $w \in$ MOD and $v \in$ INF, then $\Gamma' = \Gamma \cup \{\langle VP, uv \rangle, \langle S, uv \rangle\}$

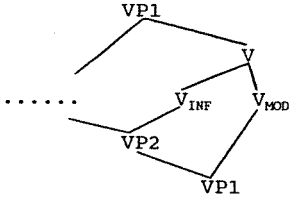
Essentially, what this rule states is the following: if we have a string that is a VP and that contains a modal verb *w*, and if certain other conditions are met, then the same string *minus w* also is a VP (and an S). In German, this modal verb *w* must be the last word of the string. In Dutch, however, the modal verb need not be the last word. Thus, in Dutch we can have the situation that there is a certain VP *xwy* such that *w* is a modal verb. In that case, the parser may conclude that the string *xy* is also a VP.

What does this mean in structural terms? The tree in (43a) depicts the situation before the application of (42) (for the German case), and (43b) represents the situation after the application of (42):

(43) a.

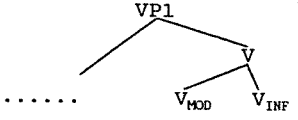


b.

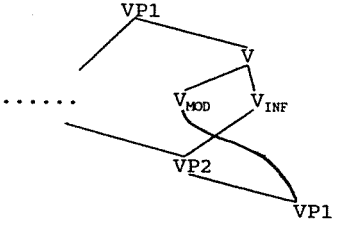


In the bottom tree of (43b) there are two VP's. VP1 contains the modal verb V_{MOD}, whereas VP2 does not. Now if we look at the situation in Dutch, a problem arises, because V_{MOD}, which itself is not contained in VP2, is enclosed by lexical material that is contained in VP2. In (44) the Dutch structures corresponding to (43a) and (b) are given:

(44) a.



b.



In the bottom tree of (44b) VP2 is a discontinuous constituent.

In the RPM formalism introduced in section 4, discontinuous constituency can be represented as follows. We can say that, in an RPM γ, the string xy forms a discontinuous constituent if $\langle X, xy\rangle \circ \gamma$ and also $\langle Y, xwy\rangle \circ \gamma$ (where X and Y are category symbols

and $x, y, w \in W^* - \{e\}$).[10] To allow for this situation we have to revise the wellformedness conditions on RPM's. This can be done by changing the definition of the notion *dominance*. To see how this works, the reader is referred to the Appendix.

The possibility of representing discontinuous constituency in this formalism allows us to state rules for the parsing of Verb Raising constructions in Dutch. These rules are the counterparts of the rules for German stated in (33). They are given in (45):[11]

(45) Let $u,s,t \in W^*$ and $v, w \in W$. Let Γ be an RPM such that $\langle V, wvx\rangle, \langle VP, uwvx\rangle \in \Gamma$.

i. If $w \in$ AUX and $v \in$ PART,
then $\Gamma' = \Gamma \cup \{\langle VP, uvx\rangle\}$

ii. If $w \in$ MOD and $v \in$ INF,
then $\Gamma' = \Gamma \cup \{\langle VP, uvx\rangle, \langle S, uvx\rangle\}$

iii. If $w \in$ CON1 and $v \in$ INF,
then $\Gamma' = \Gamma \cup \{\langle VP, uvx\rangle, \langle S, uvx\rangle\}$

iv. If $w \in$ CON2, $v \in$ INF, $u = st$ and $\langle NP, s\rangle \in \Gamma$,
then $\Gamma' = \Gamma \cup \{\langle VP, tvx\rangle, \langle S, tvx\rangle\}$

v. If $w \in$ CON3, $v \in$ INF, $u = st$ and $\langle NP, s\rangle \in \Gamma$,
then $\Gamma' = \Gamma \cup \{\langle VP, tvx\rangle, \langle S, tvx\rangle\}$

vi. If $w \in$ ACI, $v \in$ INF, $u = st$ and $\langle NP, s\rangle \in \Gamma$,
then $\Gamma' = \Gamma \cup \{\langle VP, tvx\rangle, \langle S, stvx\rangle\}$

vii. If $w \in$ RAIS and $v \in$ INF,
then $\Gamma' = \Gamma \cup \{\langle VP, uvx\rangle, \langle S, uvx\rangle\}$

Like the German rules in (33), the rules in (45) could be applied after the parser has reached the end of the sentence. As we have seen in section 5.1, however, it is also possible to apply them "on the fly" during the analysis of the sentence. The rules can be applied as soon as the verbs w and v have been identified, *i.e.* before the words making up the string x have been seen. In section 5.3 we will consider in some detail how this can be done.

---

[10] e is the empty string.

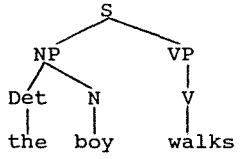
[11] Again the rules do not distinguish between infinitives with and without the infinitive marker (*te*), but they could easily be modified to do so.

5.3 *The use of string variables*

We can conceive of a parser as a machine that constructs an RPM as it goes through the sentence from left to right. So far we have only dealt with RPM's that are complete in the sense that they contain all the lexical material in the sentence. How do we deal with RPM's that are incomplete because the parser has not yet reached the end of the sentence? I will use variables ranging over strings of words (elements of W^*) which stand for that part of the sentence that the parser has not yet seen. To clarify this, I will discuss a simple example.

Corresponding to the complete tree in (46a), we have the RPM in (46b):

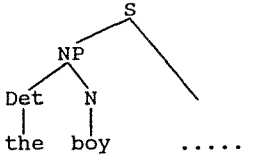
(46) a.



b. {<S, *the boy walks*>, <NP, *the boy*>, <Det, *the*>, <N, *boy*>, <VP, *walks*>, <V, *walks*>}

Now imagine that the parser has only seen the first two words of this sentence and that it knows that *the boy* is an NP and that this NP is a daughter of the root S node. In terms of phrase structure trees, what the parser has constructed so far is the incomplete tree depicted in (47a). With the help of a string variable *x* this incomplete tree can be represented as the RPM (47b):

(47) a.



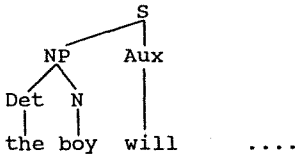
b. {<S, *the boy x*>, <NP, *the boy*>, <Det, *the*>, <N, *boy*>}

x stands for the lexical material that is still to come and that has not yet been attached to the tree. This

could turn out to be one word (like *walks* in (46)), but it could also be a string of more than one word like *never goes to bed before midnight* or *is very unhappy*. The parser doesn't know what is to come and therefore represents this unknown string of words as a variable.

At the start of the parsing process all the lexical material of the sentence is still unknown to the parser. The only thing the parser knows is that the sentence as a whole is an S. Thus the parser starts out with the singleton RPM {<S, *x*>}. In the course of the parse, the lexical items of the sentence are read one by one. When the parser encounters a word *w* of lexical category X it does two things: it adds the pair <X, *w*> to the RPM and it updates the variable standing for the unseen part of the sentence. Suppose for instance that the RPM in (47b) is what the parser has constructed so far, and that it reads the auxiliary *will*. The RPM produced by the parser is (48b), which corresponds to the tree (48a):

(48) a.



b. {<S, *the boy will y*>, <NP, *the boy*>, <Det, *the*>, <N, *boy*>, <Aux, *will*>}

In order to obtain (48b) from (47b), the parser must (i) add the pair <Aux, *will*> to (47b); and (ii) it must replace the variable *x* in (47b) by the string *will y* (where *y* is also a string variable). It is crucial to replace different occurrences of a variable by the same string. We can think of this substitution operation as the assignment of a value to the variable. Thus, in the attachment of the auxiliary to (47) the variable *x* is assigned the value *will y*.

Apart from adding lexical items to the RPM, the parser will also add higher level constituents like NP, PP and VP, as it goes through the sentence from left to right. I will assume that this facilitates comprehension, because a certain structure is imposed on the input string. The earlier the processor can recover syntactic structure, the better it is for comprehension. When the RPM is completed, *i.e.* when the

end of the sentence is reached and all the lexical material is added to the RPM, the remaining string variable is assigned the value *e* (the empty string).

5.4 *An example*

Using the method of representing incomplete phrase structure outlined above and the rules in (45), let us consider the way in which a simple Dutch Verb Raising sentence like (49) can be handled by the parser:

(49) (dat) Jan Peter het gras wil helpen maaien
that Jan Peter the lawn wants help mow
'that Jan wants to help Peter mow the lawn'

The parser starts with creating a root S-node to which the NP *Jan* is attached. I will assume that at this point the parser is able to infer that the rest of the sentence must be a VP. The RPM constructed up to this point is given in (50):

(50) {<S, *Jan x*>, <NP, *Jan*>, <VP, *x*>}

This RPM contains the following information: the string of words that is being parsed consists of two parts. The first part is the word *Jan*, the second part, *x*, is still unknown. The sentence as a whole (*i.e.* the string *Jan x*) is an S; *Jan* is an NP and *x* is a VP.

Subsequently, the next word of the sentence, *Peter*, is identified as an NP. The pair <NP, *Peter*> is added to the RPM and *x* is assigned the value *Peter y*, where *y* is a variable representing the unkown part of the sentence following after *Peter*:

(51) {<S, *Jan Peter y*>, <NP, *Jan*>, <VP, *Peter y*>, <NP, *Peter*>}

The following word is the determiner *het*. The parser knows that this is the beginning of an NP, but it does not know what other lexical material will be dominated by this NP. To handle this situation, two new variables are introduced, say *u* and *z*. *u* represents the unknown part of the new NP; *z* represents all the lexical material following after that. The pairs <Det, *het*> and <NP, *het u*> are added to the RPM and *y* is assigned the value *het u z*:

(52) {<S, *Jan Peter het u z*>, <NP, *Jan*>, <VP, *Peter het u z*>, <NP, *Peter*>, <Det, *het*>, <NP, *het u*>}

In the next step, *gras* is identified as an N and the variable *u* is assigned the value *gras*:

(53) {<S, *Jan Peter het gras z*>, <NP, *Jan*>, <VP, *Peter het gras z*>, <NP, *Peter*>, <Det, *het*>, <N, *gras*>, <NP, *het gras*>}

After this, the parser attaches the verbs *wil* and *helpen*. A new variable, *v*, is introduced and *z* is assigned the value *wil helpen v*. The parser also knows at this point that the string *wil helpen v* is a V, and, since the verbal cluster in Dutch is right branching, that *helpen v* also is a V. The pairs <V, *wil helpen v*> and <V, *helpen v*> can therefore also be added to the RPM:

(54) {<S, *Jan Peter het gras wil helpen v*>, <NP, *Jan*>, <VP, *Peter het gras wil helpen v*>, <NP, *Peter*>, <Det, *het*>, <N, *gras*>, <NP, *het gras*>, <V, *wil*>, <V, *helpen*>, <V, *wil helpen v*>, <V, *helpen v*>}

Note that at this point rule (45ii) can be applied to this RPM. If we do this, the pairs <VP, *Peter het gras helpen v*> and <S, *Peter het gras helpen v*> are added to this RPM. As a result, we get the RPM in (55):

(55) {<S, *Jan Peter het gras wil helpen v*>, <NP, *Jan*>, <VP, *Peter het gras wil helpen v*>, <NP, *Peter*>, <Det, *het*>, <N, *gras*>, <NP, *het gras*>, <V, *wil*>, <V, *helpen*>, <V, *maaien*>, <V, *wil helpen v*>, <V, *helpen v*>, <VP, *Peter het gras helpen v*>, <S, *Peter het gras helpen v*>}

Subsequently, the verb *maaien* is added to the RPM and the variable *v* is replaced by the string *maaien w*. The result is shown in (56):

(56) {<S, *Jan Peter het gras wil helpen maaien w*>, <NP, *Jan*>, <VP, *Peter het gras wil helpen maaien w*>, <NP, *Peter*>, <Det, *het*>, <N, *gras*>, <NP, *het gras*>, <V, *wil*>, <V, *helpen*>, <V, *maaien*>, <V, *wil helpen maaien w*>, <V, *helpen maaien w*>, <VP, *Peter het gras helpen maaien w*>, <S, *Peter het gras helpen maaien w*>}

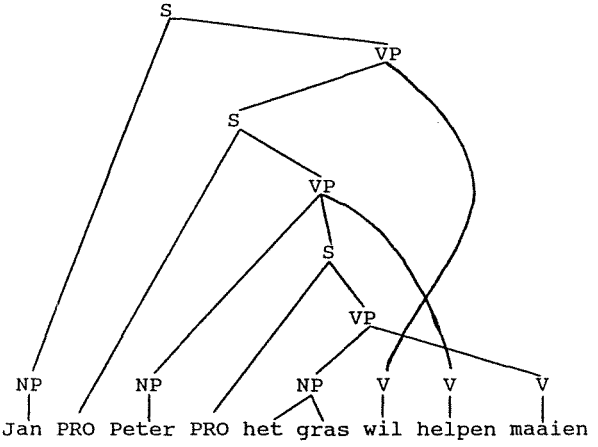
Now (45v) is applied to this RPM. Note that it can be applied in two ways: either with respect to the VP *Peter het gras wil helpen maaien w*, or with respect to the VP *Peter het gras helpen maaien w*. We must be careful to choose the second possibility. (We should impose extra

conditions on the application of the rules in (45) to ensure that the parser always takes the right VP. A possibility is to stipulate that it must always take the lowest VP, *i.e.* the VP which does not dominate another VP.) The pairs <VP, *het gras maaien w*> and <S, *het gras maaien w*> are added and as a result we get:

(57) {<S, *Jan Peter het gras wil helpen maaien w*>, <NP, *Jan*>, <VP, *Peter het gras wil helpen maaien w*>, <NP, *Peter*>, <Det, *het*>, <N, *gras*>, <NP, *het gras*>, <V, *wil*>, <V, *helpen*>, <V, *maaien*>, <V, *wil helpen maaien w*>, <V, *helpen maaien w*>, <VP, *Peter het gras helpen maaien w*>, <S, *Peter het gras helpen maaien w*>, <VP, *het gras maaien w*>, <S, *het gras maaien w*>}

The sentence is now completed, and therefore *w* is assigned the value *e*. A phrase structure tree representing one dimension of the resulting RPM is given in (58):

(58)



5.5 *Conclusion*

In this section we have seen how Verb Raising constructions in Dutch can be parsed strictly deterministically. The fact that in the course of such a parse the rules for analyzing the verbal cluster (*i.e.* the rules in (45)) can be applied "on the fly"

may explain, along the lines suggested by Bach *et al.*, why Verb Raising constructions in Dutch are easier to process than their German counterparts. In German the corresponding rules (given in (30)) can only be applied after the parser has reached the end of the sentence.

Bach *et al.* (1986) note that the contrast between German and Dutch is only found for sentences with a verb cluster consisting of at least three verbs. No contrast is found for pairs of sentences like the following:

(59) a. Jantje heeft de lerares de knikkers helpen
Jantje has the teacher the marbles help
opruimen (Dutch)
pick up

b. Wolfgang hat die Lehrerin die Murmeln
Wolfgang has the teacher the marbles
aufräumen helfen (German)
pick up help
'Jantje/Wolfgang has helped the teacher to pick up the marbles'

It is interesting to observe that this is exactly what is predicted by our analysis. In order to apply the Verb Raising parsing rule to (59a), the parser must attach both *helpen* and *opruimen* to the VP (in the same way as *wil* and *helpen* are attached to the VP in figure 11), and hence the Verb Raising rule cannot be applied earlier than in German.

Acknowledgements

I would like to thank Eric Reuland and Lyn Frazier for their comments and advice, and Bernhard Rohrbacher for helping me with the German examples. Needless to say, all errors are mine.

References

Bach, Emmon, Colin Brown and William Marslen-Wilson (1986), "Crossed and nested dependencies in German and Dutch: A psycholinguistic study", *Language and Cognitive Processes* **1**, 249-262

den Besten, H. and J. Edmondson (1983), "The Verbal Complex in Continental West Germanic", in W. Abraham (ed.) *On the Formal Syntax of the Westgermania*, John Benjamins, Amsterdam

Evers, A. (1975), *The Transformational Cycle in Dutch and German*, Ph.D. Dissertation, University of Utrecht

Haegeman, L. and H. van Riemsdijk (1986), "Verb Projection Raising, Scope, and the Typology of Rules Affecting Verbs", *Linguistic Inquiry* **17**, 417-466

Lasnik, H. and J. Kupin (1977), "A Restrictive Theory of Transformational Grammar", *Theoretical Linguistics* **4**, 173-196

Marcus, M. (1980), *A Theory of Syntactic Recognition for Natural Language*, MIT Press, Cambridge, Mass.

Appendix

A. Lasnik and Kupin (1977) define an RPM Γ as a set of monostrings (i.e. $\Gamma \subseteq W^*\Sigma W^*$) that meets the following wellformedness conditions:

(1) i. Γ contains a monostring $A \in \Sigma$.
ii. Γ contains a monostring $w \in W^*$.
iii. For every A, B ∈ Γ, either A dominates B in Γ, or B dominates A in Γ, or A precedes B in Γ, or B precedes A in Γ.

Dominance and *precedence* are defined as in (2). For a motivation and discussion of these definitions, the reader is referred to Lasnik and Kupin (1977).

(2) Let $u, v, w \in W^*$; $\alpha \in (W \cup \Sigma)^*$; $X \in \Sigma$; A, B ∈ $W^*\Sigma W^*$; A, B ∈ Γ and $A = uXw$. Let e be the empty string.
A **dominates** B in Γ iff $B = u\alpha w$, $\alpha \neq e$ and $\alpha \neq X$.
A **precedes** B in Γ iff $uvw \in \Gamma$, $B = uv\alpha$ and $\alpha \neq w$.

B. In this paper, I have adopted a modification of Lasnik and Kupin's formalism. In this version an RPM Γ is a set of pairs <X, w> such that $X \in \Sigma$ and $w \in W^* - \{e\}$ (in other words: $\Gamma \subseteq \Sigma \times (W^* - \{e\})$. The notions *dominance* and *precedence* should of course be adapted:

(3) i. If A, B ∈ Γ, then A **dominates** B in Γ iff $A \neq B$ and (for some X, Y ∈ Σ, x, y, w ∈ W^*) $A = \langle X, xwy \rangle$ and $B = \langle Y, w \rangle$.
ii. If A, B ∈ Γ, such that $A = \langle X, x \rangle$ and $B = \langle Y, y \rangle$, then A **precedes** B in Γ iff there is a C ∈ Γ such that $C = \langle Z, uxvyw \rangle$ (for some Z ∈ Σ, u, v, w ∈ W^*).

Note that according to (3i), A and B dominate each other if $A = \langle X, w \rangle$ and $B = \langle Y, w \rangle$ (and $X \neq Y$). Such mutual dominance occurs for instance in (46b) in the main text, where <VP, *walks*> dominates <V, *walks*>. (Lasnik and Kupin's definition of dominance has the same property.) This leads to a complication in the definition of direct dominance in section 4.3 (see footnote 9).

The wellformedness conditions for an RPM Γ can now be stated as in (4):

(4) i. There is an A ∈ Γ such that for all B ∈ Γ, A dominates B in Γ
ii. For every A, B ∈ Γ, either A dominates B in Γ, or B dominates A in Γ, or A precedes B in Γ, or B precedes A in Γ.

In (5) some examples are given of sets that do not meet these wellformedness requirements:

(5) {<S, *abcde*>, <B, *bc*>, <D, *cd*>} because of (ii);
{<S, *abc*>, <B, *ba*>} because of (i) and (ii);
{<A, *a*>, <B, *ab*>, <C, *c*>} because of (i) and (ii).

Note that (4) excludes partial overlap of constituents (as in the first set in (5)), and requires that the elements of Γ agree on word order (hence the unwellformedness of the second set in (5)).

C. The wellformedness conditions (4) guarantee that every RPM can be depicted as an ordinary phrase structure tree. In his analysis of Verb Raising, Huybregts introduces inconsistent RPM's that can only be represented as multi-dimensional trees. Such RPM's violate (4), but this does not mean that "anything goes". In (6), I give a definition of a wellformed multi-dimensional RPM:

(6) Γ is a wellformed **multi-dimensional** RPM iff
i. There is an A ∈ Γ such that for all B ∈ Γ, A dominates B in Γ
ii. There are $\Gamma_1,\ldots,\Gamma_n$ ($n \geq 1$) such that
a. $\Gamma = \Gamma_1 \cup \Gamma_2 \cup \ldots \cup \Gamma_n$
b. each Γ_i ($1 \leq i \leq n$) is a wellformed RPM
c. each Γ_i ($1 \leq i \leq n$) is maximal in the sense that adding an element of $\Gamma-\Gamma_i$ to Γ_i would make it un-wellformed.

The subsets $\Gamma_1,\ldots,\Gamma_n$ will be called the **dimensions** of Γ. They correspond to the dimensions of a multi-dimensional phrase structure tree. Note that (6) guarantees that the following hold:
- The dimensions $\Gamma_1,\ldots,\Gamma_n$ of Γ agree on which words occur in the sentence and what their order is. The set {<S, *ab*>, <A, *a*>, <B, *b*>, <T, *cd*>, <C, *c*>, <D, *d*>}, for instance, is still ruled out because of (6i), even though it is the union of two wellformed RPM's. For the same reason {<S, *ab*>, <A, *a*>, <B, *b*>, <T, *ba*>} is not

wellformed.
- The root of γ (the element that dominates all other elements) is a member of every dimension of γ. This follows from the maximality requirement (6iic).

D. In section 5.2 I introduced the idea of discontinuous constituency by allowing a situation in which an RPM γ contains both <X, *xy*> and <Y, *xwy*>. This clearly is a violation of (4), however, because these two pairs are neither related by dominance nor by precedence. To make discontinuous constituence possible, we either have to revise (6), or change the definition of dominance. I choose for the latter option.

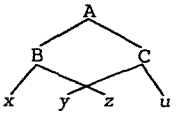
Suppose that dominance is defined as in (7):

(7) If A, B $\circ$ γ, then A **dominates** B in γ iff (for some $X, Y \circ \epsilon$ and $x_1, \ldots, x_n, x_{n+1}, y_1, \ldots, y_n \circ W^*$): A = $\langle X, x_1y_1x_2y_2 \ldots x_ny_nx_{n+1} \rangle$ and B = $\langle Y, y_1y_2 \ldots y_n \rangle$.

(7) states that A dominates B iff A contains the same lexical material as B plus some extra lexical material *which may be interspersed with the lexical material contained in B*. Thus (7) extends dominance to all cases in which the lexical material of one constituent includes all the lexical material of another constituent.

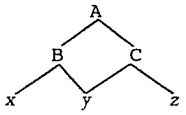
Note that, with this definition of dominance, condition (7) still rules out certain other cases of discontinuous constituency. A structure like (8), for instance, is still impossible:

(8)



The RPM corresponding to (8) is {<A, *xyzu*>, <B, *xz*>, <C, *yu*>}. In (8) A and B do not dominate each other according to (7), but neither does A precede B according to (3ii). Therefore (8) is not wellformed. (7) also rules out a structure in which two constituents partially overlap. The RPM {<A, *xyz*>, <B, *xy*>, <C, *yz*>}, which can be represented pictorially by the tree given in (9), is therefore not wellformed either:

(9)



Another RPM which is disallowed is {<A, *xyz*>, <B, *zy*>}. Here A does not dominate B although A contains all the lexical material that B contains, because the word order in A is different from that in B.

These examples show that the amended definition of dominance (7) is still quite restrictive. I want to stress however that (7) is not intended as a linguistic theory of discontinuous constituency. My purpose is to show how a limited form of discontinuous constituency can be represented in the RPM formalism. In a linguistic theory which uses this framework, universal and language specific conditions on the occurrence of discontinuous constituency will have to be formulated. In this paper, however, my concern is not with linguistic theory, but with the use of the RPM formalism for the statement of certain parsing rules.