Internal and External Remerge

On Movement, Multidominance, and the Linearization of Syntactic Objects

– revised version, August 2007 (1st version 2005)–

Mark de Vries
University of Groningen

Abstract. This article centers around two questions: (i) what is the relation between movement and multidominance/sharing, and (ii) how can syntactic structures be linearized. I claim that movement and multidominance are both instances of remerge; moreover, movement can be represented in terms of multidominance. At the PF interface, the linearization of structures involving remerge is performed by a recursive tree scanning algorithm which makes use of the inherent asymmetry between sister nodes imposed by the operation of Merge. It turns out that there are seemingly contradictory linearization demands for internal and external remerge, core examples of which are taken to be *wh*-movement and Right Node Raising, respectively; however, this can be resolved by taking into account the different structural configurations.

Keywords: remerge, movement, multidominance, linearization, PF interface, Right Node Raising

Overview

Building on the idea that a syntactic representation is derived by a sequence of combinatory operations, this article discusses the concept of remerge, and its consequences for the linearization of syntactic structures, which is a necessary procedure at the PF interface of grammar.[1] The set-up is as follows. Section 1 shows that syntactic movement can be represented in terms of multidominance. Section 2 briefly discusses the properties of the structure building operation called Merge. It is claimed that it is inherently asymmetric, which can be put to use during the linearization. Section 3 argues that remerge can be internal as well as external. Core examples are taken to be *wh*-movement and Right Node Raising, respectively. Both are represented in terms of multidominance. Section 4 discusses why remerge is problematic for the linearization of syntactic structures. It turns out that there are seemingly contradictory linearization demands for internal and external remerge. Section 5 proposes a particular linearization algorithm that handles these problems. Section 6 discusses some promising consequences of the approach. Section 7 shows what this algorithm does in terms of lists, and how complex trees can be established on the basis of a list of relations. Also, an estimate of the computational load is made, and the idea of cyclic linearization is discussed briefly. Finally, Section 8 is the conclusion.

1 The representation of displacement

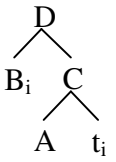
Displacement is one of the central tenets in generative grammar. The underlying idea is that a word or phrase can be related to a sentence position where it does not surface. A simple example is *wh*-movement, such as illustrated in (1a/b):

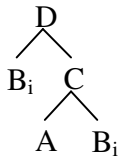
[1] I thank Jan Koster, Leonie Bosveld, Marlies Kluck, Herman Heringa, Asaf Bachrach, Roni Katzir, Eric Hoekstra, Jan-Wouter Zwart, Anneke Neijt, Anna Maria Di Sciullo, Daeho Chung, Seungwan Ha, Hans Broekhuis, Henk van Riemsdijk, and the anonymous reviewers for their comments and questions. This research was financially supported by the Netherlands Organisation for Scientific Research (NWO).

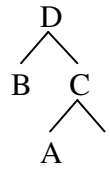
(1) a. This talented girl should purchase *a new violin*.
b. *Which violin* should this talented girl purchase _?

The unmarked direct object position in English is shown in (1a), where it is occupied by *a new violin*. In (1b) the preposed object *which violin* is thought to be related to the regular direct object position, here indicated by an underscore.

Described in these general terms, the situation is not very controversial. However, there are different views on the question how exactly displacement is to be represented or derived. Consider the possibilities in (2), rendered in graph notation:[2]

(2) a. 

b. 

c. 

d. 

In each case, B is displaced. (For ease of presentation, the distance is minimal here.) In (2a) B leaves a trace, as in Chomsky (1981). In (2b) the displaced element leaves a copy (Chomsky 1995, and subsequent work). In (2c) B is simply dislocated, leaving no trace at all (see e.g. Zhang 2004). In (2d) B is engaged in a second relationship without actual movement; this is the multidominance view on displacement, as advocated by e.g. Starke (2001), Gärtner (2002), and Frampton (2004). Put differently, syntactic movement can be compared to walking with mud under your shoes (2a), sharing of information (2b), physical movement (2c), or harpooning (2d).

The graphs in (2) are just one way of representing syntactic structures. Other common notations are bracketed structures and sets. The ordered sets corresponding to (2a-d) are given in (3a-d), respectively:

(3) a. $<_D B_i, <_C A, t_i >>$
b. $<_D B_i, <_C A, B_i>>$
c. $<_D B, <_C A, Ø >>$
d. ?

There is no straightforward set notation that matches the multidominance view in (2d). This raises the question what is real about these notations. We have to be careful not to incorporate hidden assumptions. Therefore, let us see how the structures under consideration can be derived. If we take Merge to be the basic operation that combines syntactic objects, the subsequent applications in (4) are required:

(4) a. Merge (A,B) => C
b. Merge (B,C) => D

In (4b) the constituent B is merged again. This is sometimes called 'internal merge'. If Merge is indeed a basic operation, it is clear from (4) that there is no need for additional elements like traces or copies (or even chains) for the sake of displacement *per se*. On the contrary, these can only be introduced by

[2] There are at least two more possibilities. Blevins (1990) – inspired by McCawley (1968, 1982), Sampson (1975), and others – eliminates movement by treating order as completely independent from hierarchy, including the possibility of discontinuous constituents. Koster (2007) argues against 'internal Merge', and in favor of a generalized application of pied piping; in the case of displacement, the properties of a gap are pied piped along the projection line up to the point where the relevant constituent is base-merged (and pronounced). This proposal bears resemblance to ideas current in HPSG, and related frameworks; see e.g. Sag & Fodor (1994).

additional mechanisms. Therefore, let us be minimalist and discard (2a-b/3a-b). Furthermore, notice that it is completely unclear from (2c/3c) that it is B that has been merged twice, which seems odd. This leaves us with (2d), which, in my opinion, is a very insightful way of representing movement. We have to keep in mind, however, that there is no deeper truth in this notation; what is theoretically 'real' is the derivation in (4).

2 The asymmetry of Merge

How, then, is Merge to be defined? Merge is the basic structure building operation; it combines separate syntactic objects into one new, larger object. Following general practice (that is, since Kayne 1984), I assume that this operation is binary. This assumption is not only conceptually the simplest, but it is also empirically supported by constituency tests, as far as it can be verified. Furthermore, Merge automatically creates a hierarchy: the two input objects are included in the output object. Since inclusion is equivalent to dominance, the result of Merge is that the output object created dominates the input objects.

Clearly, the output of Merge requires a new (unique) address, a label. If this were not the case, we could not operate on the result, or even refer to it. Suppose a and b are Merged, and the result is denied (or supposedly denied) a name, so it can only be referred to as the combination of its components, say a+b (and let us ignore the problem of the origin, i.e. the question where the labels of the components a and b come from). Now, if the sign + is only part of a complex naming convention, a+b *is* a label, and we might as well call the result c. If + refers to an operation of combination, it is equivalent to what Merge does, which leads to infinite regress. Thus, each projection must have a label. Whether this label is just a numeral index or an element that is predictable from X-bar theory or bare phrase structure theory is irrelevant for our purposes here.[3]

The label or projection created by Merge is indicative of a syntactic hierarchy. What about the relation between the two input objects of Merge? It is important to see that there is always some relation between them; otherwise, there would be no reason for merger to begin with. In fact, the whole point of merging, say, X and Y is to relate X and Y. Therefore, sisterhood is a basic relation in syntax: Merge creates sisters. Moreover, sisterhood is asymmetric. One could argue that one object is selected by or dependent on the other in terms of semantic or syntactic features. As a consequence, asymmetry is an inherent property of Merge (see also Koster 1999, Di Sciullo 2000, Langendoen 2003, Zwart 2004).[4] Although selection may play a role, I would like to advocate a more general perspective, in accordance with Zwart (2006) and Koster (2003). If two objects are to be merged, one is in effect added to the other. The semantic result of Merge (X,Y), where X is added to Y, is that Y says something about X. Koster calls this *relative aboutness*.[5] A simple example is the following (Koster 2003:3):

(5) a. John doesn't like beans.
b. Which book did you read?

[3] Collins (2002) argues against labels. To the extent that his arguments go through, they show that *projection* labels are not very significant for a number of syntactic processes; however, they do not show that *labels* do not exist.

[4] This view deviates from Chomsky (1995, and subsequent work) and Kayne (1994). I will return to this issue below.

[5] The use of this term is to be distinguished from Goodman's (1961), whose argument is about semantic inferences of propositions. Furthermore, it is distinct from *pragmatic aboutness* in the sense of Reinhart (1982), who defines sentence topics on that basis.

Example (5a) is a sentence about *John* [more generally: a predicate is interpreted with respect to a subject], whereas (5b) is a sentence about *books* [i.e. movement changes the salience of a constituent].

Zwart's hypothesis is that in each case, Y is syntactically dependent on X: dependency is a function of Merge. In Zwart's words: "all dependency relations are […] characterized by asymmetric sisterhood relations". Zwart focuses on subject-verb agreement. He argues that it is not established in a spec-head configuration, but rather via dependent-marking of the predicate (the sister of the subject), which is spelled out on one (or more!) of the terminals within this projection. An interesting example from this perspective is (6), a Swahili sentence taken from Carstens (2003:395):

(6) Juma a-li-kuwa a-ngali a-ki-fanya kazi
Juma SA-PST-be SA-still SA-PROG-do work
'Juma was still working.'

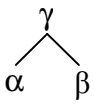
Here, SA is subject agreement, PST past tense, and PROG progressive. As is evident from the gloss, the subject agreement is spelled out not only on the finite verb, but also on other words in the predicate.[6]

The above leads to the definition of Merge in (7). The basic operation Merge establishes two primitive relations: inclusion (dominance) and aboutness (asymmetry between sisters). It should be clear that aboutness is independent from projection (if that exists).

(7) Merge $(\alpha,\beta) =_{def} \gamma$, such that
- γ includes α
- γ includes β
- β is about α

Thus, Merge creates a list of primitive relations. We can translate this list into the more usual ordered set and graph notations by the following convention:

(8) *Basic syntactic triad convention:*
(γ includes α) $\wedge$ (γ includes β) $\wedge$ (β is about α) $=_{def} <_{\gamma} \alpha, \beta> =_{def}$



Each instance of Merge creates a basic triad. A sequence of mergers gives a list of triads. A full tree or complex set can be composed on the basis of such a list. This is straightforward, with the important exception of instances of remerge, to which I will return.

Notice that it would be a misconception to equate asymmetry in syntax with linear/temporal precedence. Asymmetry between sisters does not mean that linear order is part of syntax. Nevertheless, the syntactic/semantic asymmetry can be mapped onto literal precedence at the PF interface.[7] In

[6] Admittedly, aboutness/dependency is more transparent for the relation between specifiers and their sisters (e.g. predicates), or between adjuncts and their sisters, than it is for the relation between heads and complements. I will have to leave this issue for future inquiry.

[7] By contrast – and it seems at least partly because of this confusion – Chomsky advocates a dominance-only grammar. In his notation, the result of Merge (a,b) is an unordered set {a, b}, which is then type-lifted to {a,{a,b}} in case *a* projects. This last issue is interesting in the light of the Wiener-Kuratowski convention (Wiener 1914, Kuratowski 1921), which states that an ordered pair <x, y> is equivalent to {{x}, {x, y}} (often also written as {x, {x, y}}, e.g. in Cormen et al. 1990:80; see also Quine 1945 and Schneider 1977 for comment). Although this set-theoretic reductionism has been criticized by some philosophers (see e.g. Armstrong 1986, Forrest 1986, Goodman 1986, Sider 1996), mainly because it is arbitrary, the convention is widely used. When applied to Chomsky's notation, it follows that {a,{a,b}} is in fact an ordered pair <a, b>. This is only possible if Chomsky is right that the projection label equals the head, which I think is strange from the perspective of compositionality, the intuition behind the i-within-i filter, etc. Anyway, the resulting asymmetry reflects

principle, this mapping could be performed by an intricate rule system that makes reference to language-specific parameters. However, in accordance with ideas in Koster (2003), I propose that the mapping is universal and maximally simple:[8]

(9) *Universal mapping hypothesis:*
At the PF interface, relative aboutness is mapped onto the inverse of precedence. (That is, in a basic syntactic triad $<_\gamma \alpha, \beta>$, α will precede β.)

The strongest possible hypothesis concerning the asymmetry between sisters (imposed by the operation of Merge) is this: its syntactic side (dependency), LF side (aboutness), and PF side (precedence) coincide.[9] In what follows I will use (9) as a background assumption. (Clearly, though, its implications are wide-ranging, and lead to a research program of its own, related to what was started in Kayne 1994. As the issue concerning universality is not essential for the discussion below, I will leave it at this.)

3 Internal and external remerge

The input for Merge is of course restricted to syntactic objects, but, considering general minimalist practice, we must conclude that the selection of these objects is free with respect to their location. There are three possibilities. First, input objects for Merge can be selected from the lexicon (or numeration). Second, the results of previous instances of Merge can selected as the input for Merge. These are complex objects from the syntactic work space; I will refer to them as 'partial derivations'. For example, subjects and adverbial phrases are often complex. If they are to be inserted in what can be considered as the main derivation, they must have been derived already in an independent partial derivation. Third, if there is such a thing as displacement, a constituent (term) of some partial derivation must also be accessible as a possible input object for Merge.

Thus, lexical items and/or partial derivations (all of which are roots at that point in the derivation) can be merged; this is what is usually referred to as (simple) Merge. If a constituent of some object is merged again within (actually, with) this object, we obtain Move (or 'internal Merge'). I will assume a strict cycle without much discussion. If movement can be countercyclic, basic relations of objects not directly involved in the movement process would have to be changed. For example, if A is directly related to B, and C is noncyclically moved to a position intervening between A and B, the direct relation between A and B is destroyed. Clearly, this would be an undesirable complication of the theory. A similar reasoning can be found in Chomsky (2004), who introduces the "no-tampering condition".

Crucially, movement involves *remerge*, i.e. an object that has been merged before, is merged again. If a previously merged object α is selected as input for Merge, and if the other input object is the root from which α has been selected, this instance of remerge can be called *internal*. However, this

---

the asymmetry of projection, which could be mapped onto linear order directly. However, this would only work if syntax is universally head-comp-spec (assumed by no one) or spec-comp-head, which is actually proposed by Fukui & Takano (1998), but which is in clear contrast with assumptions by Chomsky himself and Kayne (1994). See also Yasui (2004) for some discussion on related issues.

[8] Strictly speaking, not precedence but subsequence is on a par with dependency and aboutness (hence "…the inverse of…" in (9)). Since the term 'precedence' is customary, I will maintain it and ignore this minor terminological problem.

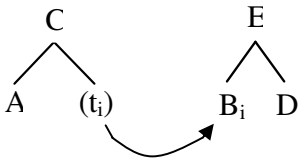
[9] The hypothesis in (9), which relates asymmetries between sisters in different domains, crucially differs from Kayne's (1994) Linear Correspondence Axiom, which, in the end, is intended to derive word order from hierarchy alone (by means of antisymmetric c-command relations, etc.). I depart from the LCA for a number of reasons. Apart from the fact that it is computationally extremely complex, implicitly denies aboutness, and introduces an asymmetry between sisters indirectly by making intermediate projections not count, the LCA is not fit for 'external remerge', to be treated in the next section.

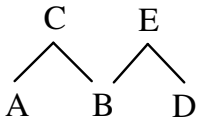
does not exhaust the possibilities: α can in principle be remerged with an independent object. This is what I will call *external remerge:*

(10) For some constituent α embedded in root R:
  a. *Internal remerge* $=_{def}$ remerge α with R.
  b. *External remerge* $=_{def}$ remerge α outside R (i.e. with some β not included in R).

The possibility of (10a/b) depends on the boundary conditions that one wishes to impose on the operation of Merge (see De Vries 2005c for more discussion). If the input for Merge is restricted to roots, remerge is excluded altogether. This point of view is defended in Koster (2007). If the familiar internal remerge is to be allowed, but the unorthodox external remerge to be excluded, specific additional conditions must be formulated (e.g. a remerged object must c-command its original position, and the output of Merge must be an independent object). It may be interesting to put off such stipulations, and allow for remerge in general (cf. Van Riemsdijk 2006).

Several possible interpretations of what can now be recognized as external remerge have been proposed in the literature. These are 'interarborial movement' (Bobaljik 1995, Bobaljik & Brown 1997), 'sideward movement' (Nunes 2001), 'upward branching' (Sampson 1975), 'multidominance/ multidomination/multiple dominance' (McCawley 1982, Blevins 1990, Wilder 1999), 'sharing' (Guimarães 2004, M. de Vries 2005b), 'grafting' (Van Riemsdijk 1998, 2006), 'parallel merge' (Citko 2005). Furthermore, multidominance is allowed in some way or another in theories involving 'parallel structures' (Williams 1978, Goodall 1987, Mu'adz 1991, G. de Vries 1992, Moltmann 1992, Grootveld 1994, Te Velde 1997). I cannot do justice to all these proposals, but the two central ideas that are relevant here are pictured in (11):

(11) a.


b.


In (11a) B is moved to an independent structure; let us call this iMove (short for interstructural movement). In (11b) B is shared between two structures; let us call this mDom (short for a hydraic multiple dominance configuration), which involves giving up the 'single mother condition' in previous frameworks. However, if structures are derived by Merge, *both* representations in (11) are derived by the following two applications of Merge:

(12) a. Merge (A,B) → C
  b. Merge (B,D) → E

In (12a) B is merged with A (which gives C). In (12b) B is remerged with D (which gives E). Since D is not related to C (the root), the step in (12b) is an instance of external remerge (cf. 10b). Thus, the perhaps surprising conclusion must be that it is only the notation that suggests a difference between iMove and mDom: without further assumptions, iMove is actually equivalent to mDom. What is relevant is that B (in this example) is engaged in two triads.

Notice that the objections against a movement notation raised in Section 1 apply to (11a) as well. For this reason, I will stick to the mDom notation in the remainder of this article. Evidently, multidominance can be used to represent internal as well as external remerge, which is a welcome

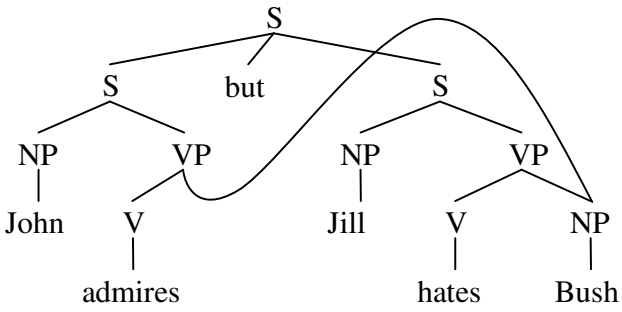
generalization. On the other hand, a set notation is not fit for external remerge. This is another argument in favor of a using a multidominance notation.

Let me provide a concrete example that probably involves external remerge. The sentence in (13) illustrates a so-called Right Node Raising configuration, i.e. backward conjunction reduction.

(13) John admires, but Jill hates Bush.

In (13) the implied object in the first conjunct is *Bush*. RNR is not restricted to NPs, but it can apply to any constituent (see the examples below). According to McCawley (1982), this construction can be analyzed by allowing a constituent to be shared between two conjuncts, i.e. in (13) the object *Bush* is dominated by both verb phrases, as is shown in (14) (my example):

(14)



The idea of applying multidominance to RNR has been picked up and defended by several authors, e.g. Ojeda (1987), G. de Vries (1992), Wilder (1999), and more recently M. de Vries (2005b), Bachrach & Katzir (2006), and Johnson (2007). Even though it is cast in different frameworks and stages of the theory, the basic idea is still the same. The main reason for treating RNR in this special way is that it behaves very differently from forward deletion (particularly, gapping); see also independent work by Neijt (1979), Wilder (1997) and Hartmann (2000). I will illustrate this with some examples from Dutch. In each case, the 'shared' constituent is printed in italics, and the position of the 'gap' is marked by an underscore. First, RNR is right-peripheral, but gapping is not; see (15):

(15) a. … dat Joop een appel _ en Jaap een peer *at*. [RNR]
… that Joop an apple and Jaap a pear ate
'… that Joop [ate] an apple and Jaap ate a pear.'

a.' * Joop _ een appel en Jaap *at* een peer.
Joop an apple and Jaap ate a pear
'Joop [ate] an apple and Jaap ate a pear.'

b. Joop *at* een appel en Jaap _ een peer. [gapping]
Joop ate an apple and Jaap a pear
'Joop ate an apple and Jaap a pear.'

Notice, by the way, that RNR is right-peripheral with respect to the coordination, not w.r.t. the sentence. This is made clear in (16) by means of coordinated subject clauses. It is a strong argument against a purely phonological treatment of RNR.

(16) [Dat Joop een boek _ en Jaap een CD *wil* *kopen*] lijkt mij onwaarschijnlijk.
that Joop a book and Jaap a CD wants buy seems me improbable
'That John [wants to buy] a book and Jaap wants to buy a CD seems improbable to me.'

Furthermore, gapping is subject to the so-called 'head condition' (Fiengo 1974), which says that if the verb is present, all its arguments must be present. This is not the case for RNR; see the contrast in (17), where the head condition is violated:

(17) a. * … dat Joop *een huis* kocht, maar Jaap _ verkocht [gapping]
… that Joop a house bought but Jaap sold
'… that Joop bought a house, but Jaap sold [a house].'
b. Joop kocht _, maar Jaap verkocht *een huis*. [RNR]
Joop bought but Jaap sold a house
'Joop bought but Jaap sold a house.'

Most importantly, gapping is subject to locality constraints, but RNR is not; see (18), which illustrates violations of the 'right roof constraint' (Ross 1967). The focus accent in (18b) is on *Anna* and *Henk*, which is indicated by capitals.

(18) a. * Piet zei dat Anna *een boek had gekocht* en Jan zei dat Marie een CD _ .
Piet said that Anna a book had bought and Jan said that Marie a CD
'Piet said that Anna had bought a book, and Jan said that Marie [had bought] a CD.'
b. [Piet zei dat ANNA _] maar [Jan riep dat Marie beweerde dat Jacob
Piet said that Anna but Jan shouted that Marie claimed that Jacob
mompelde dat HENK *een boek had gekocht*].
mumbled that Henk a book had bought
'Piet said that Anna, but Jan shouted that Marie claimed that Jacob mumbled that Henk had bought a book.'

Finally, there is a strict form identity in RNR, but not in gapping constructions. In (19) the implied verbs are indicated by a strikethrough format; here, the difference between singular and plural morphology is exploited.

(19) c. * … dat jullie boeken ~~kop-en~~ en ik CD's *koop*.
… that you:PL books buy-PL and I CDs buy:SG
'that you [buy] books and I buy CDs.'
b. Jullie *kop-en* boeken en ik ~~koop~~ CD's.
you:PL buy-PL books and I buy:SG CDs
'You buy books and I CDs.'

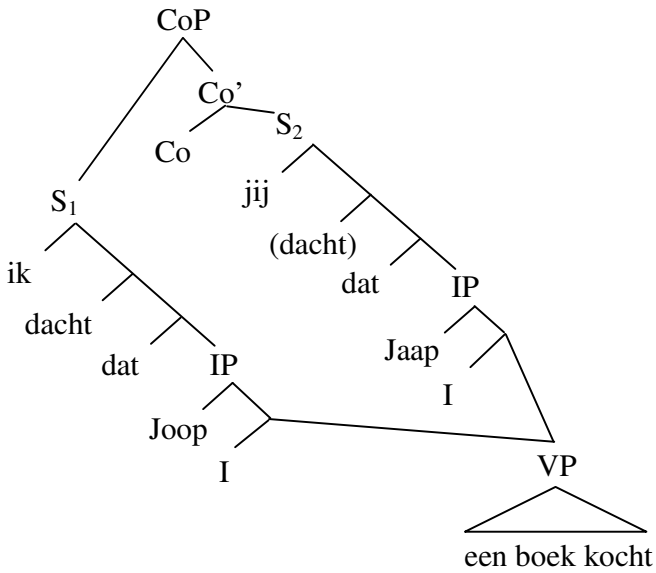
In short, if gapping (more generally, 'forward deletion' – see Wilder 1997) is deletion or ellipsis, RNR must be something else. It is also clear that RNR is not rightward across-the-board movement; moreover, Hartmann (2000) and De Vries (2005b) show that RNR is different from extraposition (contra Postal 1998).

An analysis in terms of multidominance straightforwardly explains some important properties of RNR. For instance, if there is only one (shared) constituent, morphological identity is automatically assured. Concerning the nonlocal character of RNR, consider (21), a schematic analysis of the sentence in (20).[10] Here CoP is a coordination phrase headed by *maar* 'but'.[11] (For ease of exposition I will leave out complicating details of Dutch clause structure.)

[10] The verb *dacht* 'thought' can be omitted in the second conjunct. This would be an instance of gapping. The combination of forward and backward reduction is surprisingly easy. This phenomenon is called *ambi* ellipsis in Grootveld (1994).

(20) Ik dacht dat Joop _ maar jij (dacht) dat Jaap *een boek kocht*.
I thought that Joop but you thought that Jaap a book bought
'I thought that Joop [bought a book] but you thought that Jaap bought a book.'

(21)



The shared predicate *een boek kocht* 'bought a book' is locally related to (i.e. merged with) the inflection node I in both conjuncts. Thus, multidominance creates a 'bypass'; therefore, the complexity of the conjuncts themselves is irrelevant.

In conclusion, there are two types of remerge: internal and external. Examples of these are *wh*-movement and Right Node Raising, respectively. Since multidominance (external remerge, to be precise), is not generally accepted, I briefly defended a concrete application, namely the case of Right Node Raising. It seems to me that this is the most convincing example of external remerge (of course I may be wrong; different analyses have been advanced), but other candidates figure in the literature, e.g. transparent free relatives, head movement, parasitic gaps, and across-the-board movement. Each of these constructions will have to be subject to close scrutiny. However, what is of interest here is not so much the analysis of these particular constructions, but the general mechanism of external remerge. I have shown that both internal and external remerge can be represented in terms of multidominance. This raises questions for the linearization of these objects, which is the subject of the next sections.[12]

---

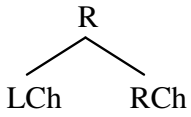
[11] Actually, I prefer a specific parallel structure approach to coordination, along the lines of De Vries (2005a), where Co' (including the second conjunct) is put 'behind' the first conjunct by means of a second type of Merge, which is used for parataxis and parenthesis in general. It should be clear, however, that the issue of '3D-coordination' is in principle independent from the matter of multidominance. Nevertheless, notice that the problem of crossing branches is voided in a 3D-representation.

[12] It should be noted that Wilder (1999) attempts to adapt Kayne's LCA in order to deal with a sharing approach of RNR (see also Johnson 2007 for some discussion). His theory is incapable of generalizing over internal and external remerge, though. Citko (2005) argues that sharing is only possible if the shared constituent is ATB-moved because it is necessary that it reaches an unequivocal position for spell-out. Citko's approach is even more limited than Wilder's; she allows for sharing, but cannot treat RNR (or simple movement) in these terms.

4 Linearization issues

A syntactic representation can be derived on the basis of a sequence of mergers. As long as there is no instance of remerge, the linearization is more or less straightforward – that is, from a linguistic perspective. In general, things are somewhat more complicated. A standard (order-sensitive) top-down tree traversal produces a string of node contents. First consider the basic triad in (22), were R is the root, LCh the leftmost child, and RCh the rightmost child.

(22)



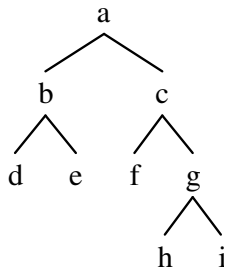
There are three possibilities: (i) the 'preorder' traversal, which lists the root first, and then the children from left to right; (ii) the 'postorder' traversal, which lists the children before the root; and (iii) the 'inorder' traversal, which lists the leftmost child first, then the root, and then the rightmost child.

Tree traversal is a recursive algorithm, consisting of three basic steps (in a binary tree): scan the leftmost child, scan the rightmost child, and perform some action, such as listing the present node content. The core of this procedure is stated in (23); here the three possible positions of undertaking the action are indicated.

(23) *scan*
↓ ← list present node [preorder]
↓ scan left child
↓ ← list present node [inorder]
↓ scan right child
↓ ← list present node [postorder]

The results of scanning the more complicated abstract tree in (24) are given in (25).

(24)



(25) preorder: [a b d e c f g h i] ('spell out before going down')
inorder: [d b e a f c h g i] ('spell out before going down the second time')
postorder: [d e b f h i g c a] ('spell out before going up')

However, a linguistic linearization requires a list of end nodes ('terminals') only. In (26) the strings from (25) are repeated, with the terminals printed in boldface.[13]

[13] An interesting alternative is proposed in Yasui (2002, 2004), who defines syntactic structures without projection nodes (e.g. [$_{will}$ it [$_{be}$ raining]] 'it will be raining'). Different ways of scanning such structures produce different word orders.

(26) preorder: [a b **d e** c **f** g **h i**]
inorder: [**d** b **e** a **f** c **h** g **i**]
postorder: [**d e** b **f h i** g c a]

The required ordering of terminals [d e f h i] is obtained in each of the three cases. Thus, ordering terminal nodes is much less arbitrary than ordering projection nodes. It seems to me that this is a welcome conclusion. An algorithm that produces a string of terminals is given in some more detail in (27):

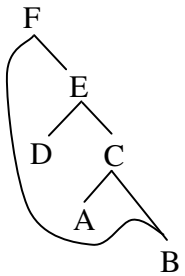
(27) a. algorithm *linearization*
create a new, empty string
select the root node
scantree
end
b. procedure *scantree*
if the present object (directly) includes members
then select the preceding member
scantree
select the other member
scantree
else add the present object to the string
return

Here, a node is only added to the string if it does not include other nodes. So *scantree* is recursively called upon without any action until a terminal node is reached (cf. Kremers 2007).

Let us work out in detail what happens if we linearize the tree in (24). We start at the root, which is *a*. This node has members; we turn to the preceding one, *b*, first; *b* is complex as well; we turn to child *d*, which does not include members and is therefore added to the string. We return to *b* and scan the rightmost child *e*, which does not include members; hence it is added to the string. We return to *a* (via *b*) and scan the rightmost child *c*, which is complex; we turn to child *f* and add it to the string. We return to *c* and start scanning the rightmost child *g*, which is complex. We scan child *h* and add it to the string, return to *g*, scan the rightmost child *i* and add it to the string. We return (four times) and end the algorithm. The string produced is [d e f h i], as required.

As stated before, the possibility of remerge complicates the linearization. Consider the case of internal remerge first. An abstract example is provided in (28), represented in terms of multidominance:

(28)



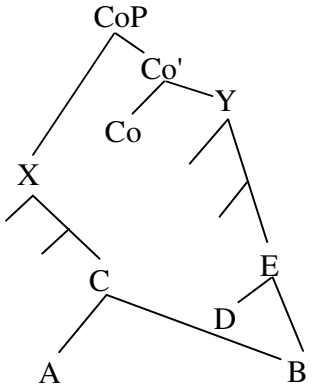
The constituent called B is displaced. The intended spell-out is [B D A]. However, when applying the algorithm in (27), we encounter B twice, and the result would be [B D A B]. Therefore, consider the condition suggested by Frampton (2004):

(29) The linearization of α (or β) in $\overset{\gamma}{\underset{\alpha}{/}\backslash_{\beta}}$ is omitted if α has a parent outside γ.

Here *outside* means 'not dominated by'. The effect is that α is pronounced in its highest position. In (28) the relevant node is B, whose two parents are F and C. When we arrive at B directly from F, there is no parent outside F, hence the linearization is *not* omitted. If we arrive at B from C, there is another parent outside C, namely F; therefore, the linearization is omitted the second time. As a consequence, B is pronounced before D and A.

So far, so good. But now consider the case of external remerge (which is not discussed by Frampton). An abstract example that corresponds to a Right Node Raising configuration is provided in (30):

(30)



The intended ordering of terminals is [A Co D B]. Let us see what happens if we apply Frampton's condition. The relevant node is B again, which has two parents, C and E. When we arrive at B from C, we have to check if there is another parent that is not dominated by C. This is the case, as E satisfies the condition; therefore the linearization of B is omitted at this point. Later, when we arrive at B from E, we determine that there is another parent that is not dominated by E, namely C, and again B is not linearized, although it should be. Thus, B will not be spelled out at all.

In short, internal remerge requires that a node with more than one parent is spelled out in the *first* position accessed, whereas external remerge requires that a node with more than one parent is spelled out in the *last* position accessed. Thus, we face a nontrivial problem. However, it can be overcome if the different configuration created by internal and external remerge is taken into account. This is the subject of the next section.

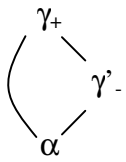
5 A conditioned algorithm for the linearization of syntactic structures

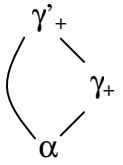
In the case of internal remerge (28) one of the two parents of the 'shared' node B is (indirectly) included in the other parent. By contrast, in the RNR configuration in (30) the two parents are not related in such a way. How, then, can the spell-out of B be regulated? The answer seems to be that we have to keep track of which nodes have been scanned. The first time B is encountered (i.e. coming from C), the other parent (E) is still unscanned. The second time B is encountered (coming from E), the other parent (C) has already been scanned. Thus, an extended version of the condition in (29) can be drawn up as follows:

(31) The linearization of α with parent γ is omitted if (i) *and* (ii):
  (i) α has another parent γ' outside γ [i.e. ∃γ', γ'≠γ: γ' directly includes α ∧ ¬ γ includes γ']
  (ii) *either* γ' includes γ *or* γ' is not yet scanned.

Implicitly, the condition in (31) requires that you know where you came from (i.e. γ), when encountering α. This, of course, is meta-knowledge that a simple scanning algorithm does not possess, at least not in a direct way. Therefore, consider the possible configurations in some detail from the perspective of a scanning algorithm that traverses a tree structure and marks each node as [+scanned] at the moment it becomes active (i.e. when it is encountered).

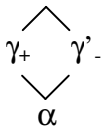
In (32) the situation for internal remerge is illustrated. Here, α is the shared node; γ and γ' are the two parents, which are in a domination relation. For ease of exposition, all other sentence material is omitted. (32a) and (32b) represent two different stages during the scanning procedure. The subscript + or - on $\gamma^{(')}$ indicates whether this node has already been scanned or not. In each case, α is the active node.

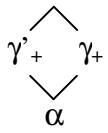
(32) a. 

b. 

In (32a) we arrive at α directly from γ; the 'other parent' is γ'. The algorithm does not know this, but it can establish that α has two parents, and that only one of them is [+scanned]. (See the next section for more details on the issue of recognizing parents in general.) At a later stage during the linearization, α is encountered the second time; this is the situation in (32b). Here, we arrive at α directly from γ, and γ' is the 'other parent', which has already been scanned before. Clearly, which of the two parents is the 'other parent' is relative to the situation. The algorithm does not know this, but what it can detect is that now both parents are marked as [+scanned]. Since (32) represents internal remerge (movement), the situation in (32a) is the one where α is to be spelled out; it is to be omitted in (32b).

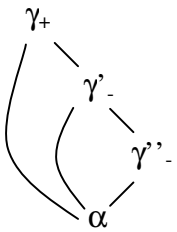
The case of external remerge is illustrated in (33). The shared node α is dominated by two parents γ and γ', which are not in a dominance relation (but still embedded under one root, at least in the case of RNR).

(33) a. 

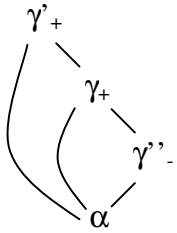
b. 

The picture in (33a) represents the situation the first time α is encountered, (33b) the second time. As was shown in the previous section, α has to be omitted the first time, and spelled out in (33b). Since the relation between α and the two parents is no different in (33) than in (32), the relation between the two parents themselves has to be taken into account. That is, the algorithm will have to check whether the parents are in an inclusion relation or not.

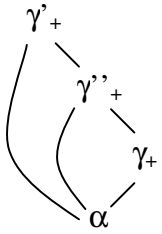
At this point, consider what happens if there are more than two parents. The pictures in (34) represent the case of internal remerge applied more than once, i.e. successive-cyclic movement via an intermediate landing site.

(34) a. 

spell out α

b. 

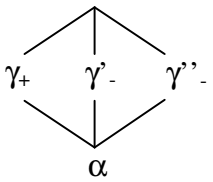
omit α

c. 

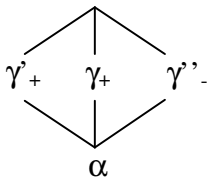
omit α

The pictures in (36) represent multiple external remerge. A concrete Right Node Raising example would be (35):

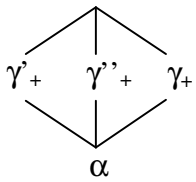
(35) John admires _, Bill hates _, and Mary loves *Bush*.

(36) a. 

omit α

b. 

omit α

c. 

spell out α

From these configurations it becomes clear that it is computationally more straightforward to change perspective by treating omission as the default, and checking if spelling out is necessary. Furthermore, the conditions for spell-out can be made concrete by counting the number of scanned parents. In (34), α is spelled out only if one parent is [+scanned]. In (36), α is spelled out only if all parents are [+scanned]. (Of course the same is valid for (32) and (33), respectively.)

The complete linearization algorithm is stated in (37) below, supplemented with some comments between curly braces. Notice that the simple algorithm from (27) in the previous section functions as the core in lines 1-11 plus 19. Line 12 verifies if the present terminal node has one parent (the 'normal' case); if so, it is added to the string in line 13. Lines 14-18 cover the case of remerge/multidominance; 'normally', these are skipped over.

(37) algorithm *linearization* {performs the conditioned linearization of the terminals of a binary tree with the possibility of multidominance}

1 create a new, empty string
2 select the root node
3 *scantree*
4 end

```
5   procedure scantree                                   {recursive depth-first scanning procedure}
6   mark the present object as "scanned"
7   if the present object includes members
8       then select the preceding member                                        {else terminal}
9            scantree
10           select the other member
11           scantree
12      else if the present object has one parent
13               then add the present object to the string                      {simple terminal; else remerged}
14               else if there is a parent that includes every other parent     {internal remerge}
15                        then if the number of parents that is marked as "scanned" equals 1
16                                 then add the present object to the string    {else do nothing}
17                        else if every parent is marked as "scanned"           {external remerge}
18                                 then add the present object to the string    {else do nothing}
19  return
```

The conditioned spell-out in lines 14-18 reflects the intention of (31), elaborated in the way explicated above. Lines 15-16 apply to the situation caused by internal remerge; lines 17-18 to the configuration that is the consequence of external remerge. The spell-out in lines 16 and 18 is conditional; the default is to perform no action, which results in the omission of the linearization of the shared terminal node at the relevant position.

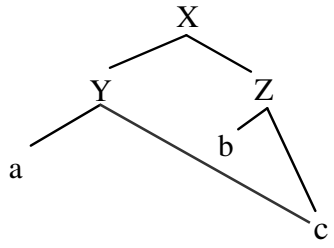
Thus, the linearization algorithm in (37) captures the difference between internal and external remerge for linear order, whilst preserving the generalization that both simply involve remerging some constituent, which is represented in terms of multidominance (but see section 7).

6 Extending the approach to more complicated cases

The previous section addressed the linearization of structures involving multidominance as the result of either internal remerge or external remerge, covering the empirical cases of simple movement and successive cyclic movement on the one hand, and right-hand sharing constructions exemplified by Right Node Raising and multiple RNR on the other hand. Here, I will discuss some consequences of the present approach for other and even more intricate constructions.

Merge is usually assumed to obey the strict cycle, or extension condition (Chomsky 1995: 190, 327). That is, the derivation always proceeds at the root level. As a consequence, movement to an embedded position is impossible, and substructures cannot be tampered with. I fully comply to this principle, which means that primitive syntactic relations, once established, cannot be altered or destroyed. In fact, a considerable complication of the theory would be necessary to overcome the strict cycle, which follows automatically from the way Merge is presently defined. Nevertheless, we established that the possible use of external remerge in a derivation may eventually lead to a structural representation that *apparently* involves a violation of the strict cycle. Consider (38):

(38)



This abstract representation can be derived cyclically by first merging c with a, then externally remerging c with b (thereby creating a second root node Z), and finally merging the two roots, Y and Z. The derivation of (38) necessarily involves external remerge, and this has consequences for the linearization; applying the algorithm in (37) gives the string of terminals [a b c] (and not [a c b]). Crucially, therefore, it is impossible to analyze c in the representation of (38) as being moved (in the traditional sense) to the embedded position within Y, which is a desirable result.[14] Instead, (38) is a sharing construction of which Right Node Raising like in (13) or (21) was presented as an example.

It does not automatically follow from the approach that (38) is only possible in coordinative constructions (that is, where the node joining the double-rooted substructure is a CoP). Of course we could stipulate such a limitation as a syntactic constraint, but it seems to me that this is not the most interesting way to go. Let us explore the syntactic possibilities predicted by the present proposal without additional conditions.

First, RNR-like behavior can be found in constructions that involve syntactic subordination. Some examples in Dutch are given in (39):

(39) a. Het kan moeilijk zijn om syntactische _ van semantische *factoren* te onderscheiden.
it can difficult be to syntactic from semantic factors to distinguish
'It can be hard to distinguish syntactic _ from semantic *factors*.'

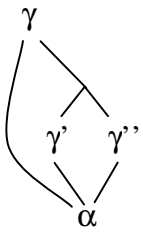
b. Joop is, ofschoon een schuldbewuste gebruiker van _, niettemin principieel
Joop is although a contrite user of nevertheless principally
gekant tegen *de intracontinentale luchtvaart*.
opposed against the intracontinental aviation
'Joop is, although a contrite user of, nevertheless principally opposed to intracontinental aviation.'

This possibility has been noticed before in Huybregts & Van Riemsdijk (1985), among others. Examples as in (39) seem to confirm the idea that semantic coordination and syntactic coordination may be independent from each other; see Culicover & Jackendoff (1997) and Van der Heijden (1999) for interesting discussion.

Next, consider the complicated situation of a combination of internal and external remerge. There are two basic possibilities. The first is pictured in (40). It involves internal remerge of a constituent that has already been externally remerged.

[14] In this paper, I will ignore the issue of head movement. Notice, however, that an analysis in terms of syntactic adjunction is incompatible with the present proposal. I do not take this to be a disadvantage; rather, I would favor the idea that head movement results in fusion.

(40)

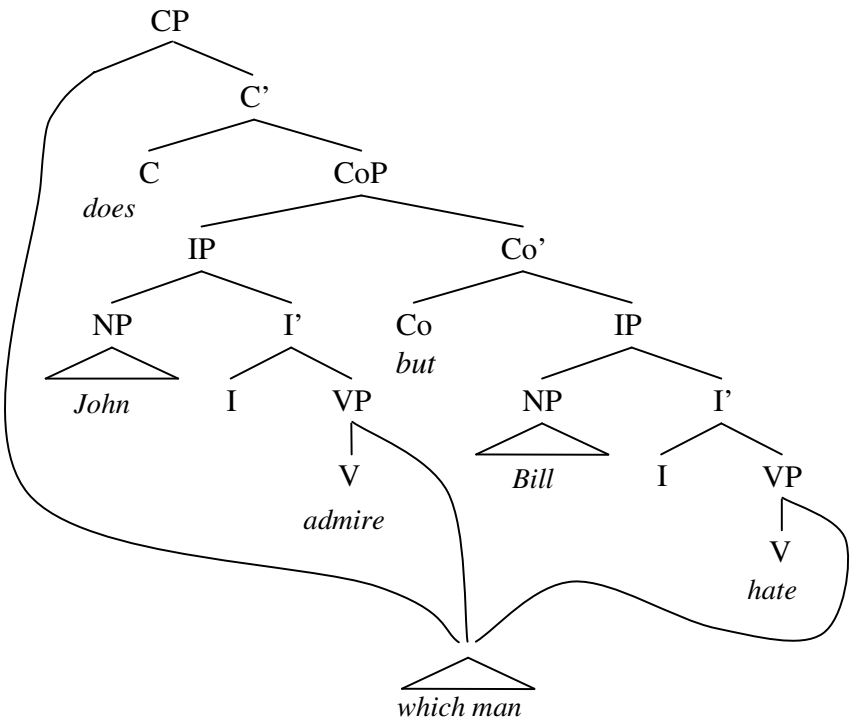


A concrete application of this structure might be across-the-board movement, such as illustrated in (41). Here, the *wh*-moved object is related to both conjuncts. The idea of treating ATB in terms of sharing has been advanced by Williams (1978), Goodall (1987), and others (most recently Citko 2005).

(41) *Which man* does John admire _ but Bill hate _?

Since there is one parent, γ, that includes all other parents of α in (41), namely γ' and γ'', the structure is treated on a par with a standard movement configuration by the algorithm in (37). Therefore, α – e.g. *which man* in (41) – is spelled out in the first position accessed, i.e. the highest position, as required. A more detailed tree structure for (41) is provided in (42), which abstracts away from subject movement, etc., for ease of representation:

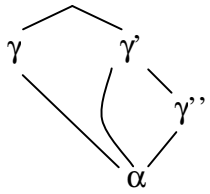
(42)



The object *which man* is shared between two conjuncts; moreover, it is moved out of the coordination phrase to SpecCP, where it is pronounced.

The second option is illustrated in (43). It involves internal remerge followed by external remerge.

(43)



There is no parent that includes all other parents; therefore, α will be spelled out in the last position accessed by the algorithm in (37). A possible interpretation of this is the following: in a RNR context, it is pointless to move the shared element within one or more conjuncts, as it will always be spelled out in the final position. This perhaps unexpected result is empirically correct: RNR is right-peripheral (the name says it all), as noted before. Movement (of the element to be shared) *within* the coordination is theoretically possible, but it will be covert. By contrast, movement *out of* the coordination phrase can be overt, because it leads to a radically different configuration, namely (40), which represents ATB. Interestingly, there is no periphery condition restricting ATB, witness (44) in Dutch:[15]

(44) *Wat* heeft Joop _ gedaan, maar Jaap _ nagelaten?
what has Joop done but Jaap refrained.from
'What did Joop do, but Jaap refrain from?'

The possibility of (44) simply follows from the present approach. This fortifies the decision to refrain from hastily building in additional constraints into the linearization algorithm (37), which operates on structures that may involve external remerge in general. As a consequence, the particular right-periphery condition for RNR must be explained in another way. In fact, this specific conclusion is corroborated by Kluck (to appear), who shows that purely syntactic approaches to derive the periphery effect fail; instead she proposes to combine Hartmann's (2000) theory on the semantics and prosody of the RNR construction with a multidominance approach.

I conclude that the proposal laid down in the previous sections makes a number of interesting predictions concerning cyclicity, and the combination of internal and external remerge. We have seen that not only ordinary RNR, but also subordinative sharing and ATB can be analyzed in terms of external remerge.

7 Syntactic representations, lists of local relations, and the cycle

In the previous three sections, I have ignored one important issue. The linearization has been worked out on the basis of multidominance trees, which are representations based on a sequence of mergers. Merge itself has been defined in terms of the primitive relations inclusion and aboutness in Section 2. Therefore, consider how exactly these representations are brought about. Furthermore, I will describe what the linearization algorithm (37) does in terms of a list of local relations. In particular, it remains to be shown how the conditions in (37), lines 7, 12, 14, 15, and 17, are checked. Finally, the interaction with syntactic cycles is discussed briefly.

[15] Recall that Dutch is overtly SOV (modulo V2). Note that I am not claiming that the base order is OV (arguments for a VO basis are provided in Zwart 1994), only that ATB operates on derived structures. The same has been shown for RNR, which can be fed by heavy NP shift; see Wilder (1997) and Bachrach & Katzir (2006).

7.1 The construction of (multidominance) trees

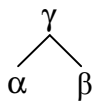
The Universal mapping hypothesis (9) states that relative aboutness is mapped onto the inverse of precedence. Therefore, the primitive relations defined by Merge (7) that are transmitted by the PF interface are inclusion and precedence:

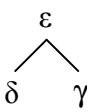
(45) Merge $(\alpha,\beta) \rightarrow \gamma \equiv \gamma$ includes $\alpha \wedge \gamma$ includes $\beta \wedge \alpha$ precedes β.

According to the Basic syntactic triad convention (8), (45) is equivalent to the minimal tree  . Suppose the following two instances of Merge are performed:

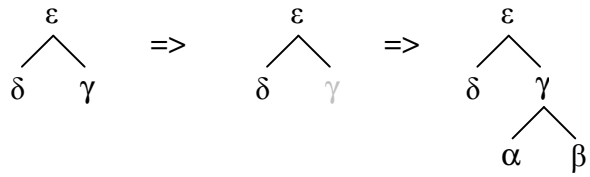
(46) a. Merge $(\alpha,\beta) \rightarrow \gamma$
b. Merge $(\delta,\gamma) \rightarrow \varepsilon$

This gives the lists of relations and the basic triads in (47):

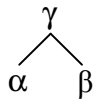
(47) a.  γ includes α, γ includes β, α precedes β

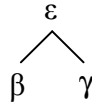
b.  ε includes δ, ε includes γ, δ precedes γ

The complete list is obtained by simply adding the two lists together. The complex tree representation is obtained by substituting γ in (47b) by (47a). This is shown in (48). How do we know that γ is to be substituted? The reason is of course that it is present in more than one triad, that is, γ both includes and is included.

(48) 

Next consider internal remerge. In (49) β is moved.

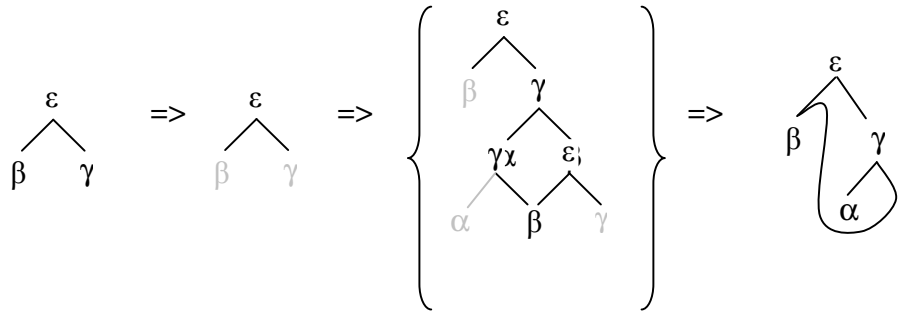
(49) a. Merge $(\alpha,\beta) \rightarrow \gamma \equiv$  γ includes α, γ includes β, α precedes β

b. Merge $(\beta,\gamma) \rightarrow \varepsilon \equiv$  ε includes β, ε includes γ, β precedes γ

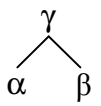
As in (48), γ in (49b) is substituted. But now β also functions in two basic triads: it is not only included in ε, but also in γ. Therefore, β must be redefined as well. However, note that the type of substitution

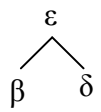
that is necessary for β is different from the one needed for γ (typographically it involves growing upward instead of downward). The order of substitutions is irrelevant; (50) shows substitution of γ first next to substitution of β first – the result is the same:

(50)

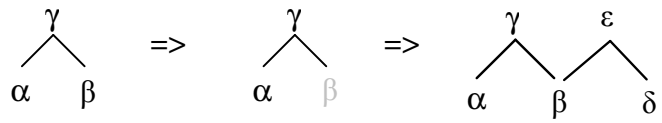


Of course the intermediate representations do not exist; they are just construction steps.[16] For completeness sake, the case of external remerge is given in (51) and (52):

(51) a. Merge (α,β) → γ ≡  γ includes α, γ includes β, α precedes β

b. Merge (β,δ) → ε ≡  ε includes β, ε includes δ, β precedes δ

(52)



In the next subsection I will return to the linearization algorithm.

7.2 Linearization in terms of lists and basic triads

In formalized grammars it is often the case that a designated node is defined as the root node. In principle, this is unnecessary, because it can be derived which node is the root node by going over the complete list of primitive relations (just once). For instance, the list in (53), which corresponds to (46)-(48), can only be interpreted such that ε is the root: it is the only node that is not included by any other node.

(53) γ includes α, γ includes β, α precedes β, ε includes δ, ε includes γ, δ precedes γ

If we find more than one node that satisfies this criterion, the list is not linearizable. For example, the list in (54), which corresponds to the hydraic structure in (52), produces two possible roots: γ and ε.

[16] In the second possibility between braces, β is replaced by a larger unit. From the list we know that γ includes β. However, if one calculates with basic triads as the measurement unit, the other child of γ, α, might as well be added right away (this is shown in graytone). Again, the result is the same.

(54) γ includes α, γ includes β, α precedes β, ε includes β, ε includes δ, β precedes δ

Thus, (54) would crash at the PF interface, contrary to a full RNR or ATB construction, in which a uniting instance of Merge has taken place (e.g. in (30) or (42) above).

When there is a detectable root, the linearization procedure *scantree* can be started. It is repeated here for ease of reference:

(55) procedure *scantree*

```
6     mark the present object as "scanned"
7     if the present object includes members
8         then   select the preceding member; scantree
10                select the other member; scantree
12        else if the present object has one parent
13                then add the present object to the string
14                else   if there is a parent that includes every other parent
15                              then if the number of parents that is marked as "scanned" equals 1
16                                        then add the present object to the string
17                              else if every parent is marked as "scanned"
18                                        then add the present object to the string
19    return
```

In line 7, it is checked if the active node (say *AN*) includes members. This condition is fulfilled if the list of relations contains the items *AN includes Ch1* and *AN includes Ch2*. (Note that the binary character of Merge makes sure that there will always be two such statements.) In order to select the preceding or the other child (line 8 or 10) it is necessary to go over the list again and find the item *Ch1 precedes Ch2* or the other way around.

In line 12 it is checked if the present node has one parent. This amounts to going over the list and find all items of the type *x includes AN*. The parents $\{x_1, \ldots, x_n\}$ ($n \geq 1$) are temporarily stored. If there is just one such statement (i.e. the cardinality of the set of parents is one), the condition is fulfilled. If not, we reach line 14, which checks if there is a parent that includes all other parents. Computationally, this is the most costly condition. A straightforward way to proceed is as follows, for each parent x_i separately. Go over the list and check for items of the form x_i *includes* y_j. Temporarily store all y_j in a set. If this set is not empty, go over the list again and check for items of the form y_j *includes* z_k. Temporarily store all z_k in a set. If this set is not empty, go over the list again, etc. Finally, check if all other parents $x_{o,\, o \neq i}$ are present in the unified set of dominated nodes $\{y_1, \ldots, y_{n(y)}, z_1, \ldots, z_{n(z)}, \ldots\}$. If so, the condition is fulfilled. For instance, if there are four parents, it could be that the set of (grand)*children of x_2 include y_3, z_2, and w_6, which happen to equal x_1, x_3 and x_4. In that case, there is a parent that includes all other parents. Finally, the conditions in lines 15 and 17 simply require a look at the list of parents (which usually has only two members).

What is the computational load of linearizing a syntactic object by a scanning algorithm? For any object, the number of nodes and the number of items in the list of basic relations is proportional to the number of mergers performed. For each node, the procedure *scantree* is passed through. In turn, this involves going over the list two times (lines 7-12), plus – in case there has been remerge – going over the list, looping over the tree depth times the average number of parents. Thus, we obtain the formula in (56), which shows polynomial growth:

(56) $CS = c_0 + c_1 \cdot M + c_2 \cdot M^2 + c_3 \cdot M^3$

Here, CS is the number of computational steps, M is the number of mergers involved in the derivation, and c_0 through c_3 are constants. The term $c_1 \cdot M$ involves finding the root; $c_2 \cdot M^2$ is necessary for

scanning the tree and finding the terminals; $c_3 \cdot M^3$ is the toll for allowing remerge/multidominance.[17] Note that c_3 is relatively small, since (i) most probably, only a minority of nodes is remerged, and (ii) the tree depth is only a fraction of the number of nodes.[18] It seems to me that the moderate, subexponential character of (56) is acceptable.

The estimate above is a worst case scenario, in which all the information that is necessary for the linearization is calculated at the PF interface. It is also conceivable that each node α carries information such as a complete set of nodes included by α. A set like this can be established by Merge during the derivation, simply by adding the 'inclusion sets' of the two relevant input nodes.[19] With this information available, (56) would reduce to linear growth: $CS = c_0 + c_1 \cdot M$, where c_1 is relatively large.

7.3 A note on syntactic cycles

We have discussed the linearization of completed syntactic derivations. However, it has been suggested that syntax operates in cycles or phases (see Chomsky 2004 in particular). Here, I will briefly address possible consequences of this idea.

Linearization is a top-down procedure, necessarily starting at the root. By contrast, the syntactic derivation is bottom-up. At the end of each syntactic cycle, the structure will be handed over to the interfaces (for our purposes, only PF is relevant), and the substructure will be linearized. When the next cycle is entered, the previous cycle becomes opaque (ideally, in every respect). As long as there is no remerge, this is indeed possible. In fact, for each step of Merge $(\alpha, \beta) \rightarrow \gamma$ it is the case that *linearization* (γ) = *linearization* (α) + *linearization* (β). Suppose β is a cycle, then the outcome of linearization (a string) is already stored, and the structure of β need not be scanned again. If, however, remerge is at stake, nontrivial problems arise.

Per definition, internal remerge across a cycle boundary is impossible, unless the designated escape hatch (the edge) of the phase is used. For external remerge we can simply state that it must take place before the cycle is closed (giving rise to a hydraic structure, as discussed). Let us restrict the discussion to internal remerge for the moment. Suppose some object δ has been remerged (that is, 'moved' to the edge). It is normally assumed that the edge of a cycle is not passed on to the interface (for the obvious reason that what is in the edge will be used in the higher cycle).[20] But then the question arises how the spelling out of δ within the lower cycle can be prevented. Its linearization should be omitted (in Frampton's words), but the information that δ has another parent, which would lead to this decision, is not present at the interface during the spelling out of the lower cycle.[21] (Unfortunately, Frampton 2004 does not discuss this issue, and neither does Chomsky.)

To be a little more concrete, consider the following (simplified) example. In (57a), C is the phase head, and *wh* is internally remerged to the edge (hence must not be spelled out in its lower position).

---

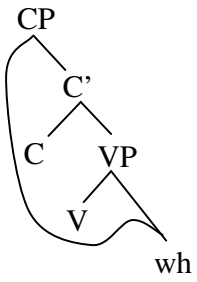
[17] From a different perspective, Karttunen & Kay (1986) warn that the amount of computational effort that goes into producing copies is much greater than the cost of "unification" [i.e. multidominance] when a tree is being parsed. For this reason, they advocate structure sharing.

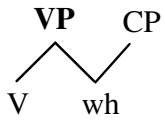
[18] Potentially, a syntactic tree is surface filling. In that case, the tree depth is proportional to the square root of the number of nodes. This would reduce the term $c_3 \cdot M^3$ to a fractional power $M^{2.5}$. However, in reality syntactic trees are not like this: the depth of the (sub)constituents of a sentence are not (necessarily) proportional to the depth of the major projection line.

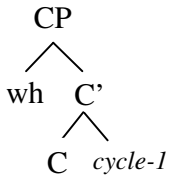
[19] Similarly, it is possible to establish c-command relations during the derivation, as is proposed by Epstein (1999).

[20] A different take on this issue is advanced in Fox & Pesetsky (2005). I will refrain from reviewing their proposal (which relates to data concerning object shift), but see, e.g., Nilsen (2005) for comments.

[21] Of course traces (in the GB sense), unlike copies or shared nodes, contain such information per definition, but they violate Chomsky's Inclusiveness Condition (compare the discussion in section 1).

(57) a. 

b. cycle 1 

c. cycle 2 

The spelling out procedure would have to be as follows: i) if a phase head is selected, its edge must be filled first (here, creating the CP level); ii) after that, the complement of the phase head (here, VP) is sent to the interfaces; iii) the linearization procedure at PF must also be informed if any node included within this complement is related to the edge (here, *wh*), and if so, what the relevant root of the resulting hydraic structure is. At the end of the first cycle, then, the linearization algorithm will be confronted with (57b), which yields the string [V]. Even though there is no second phase head triggering spell out, the remaining material must be linearized, so there will have to be a second cycle, as is indicated in (57c), which results in the string [wh C + *cycle-1*] = [NP C V]. Clearly, then, the introduction of cycles causes considerable complications for the linearization procedure. Notice that this is independent of the question whether multidominance or copies are used in the representation. What I find most disturbing is that remerge information is necessarily used twice (in the example above, the fact that CP includes *wh* is 'known' to both cycles).[22] By contrast, if the linearization does not operate in cycles – as we have assumed so far –, all these problems are avoided.

I conclude that if remerge exists (whether internal or external), linearization is hard to combine with the idea of syntactic cycles or phases.[23] The reason is essentially that the former operates top-down, but the latter bottom-up. Thus, if syntactic cycles exist – and at least for finite CPs this is not very controversial –, linearization must be exempt from its restrictions; this means that the full syntactic structure must be stored, and made accessible to the linearization procedure after the completion of the full sentence.

8 Conclusion

Complex syntactic objects are derived by an operation called Merge. The input for Merge is free in the sense that input objects can be selected from the lexicon (or numeration), from the syntactic work space (i.e. partial derivations that are the output of earlier instances of Merge), and from within partial derivations. The last option leads to the effect that an item can be merged more than once, in other words, *remerge*. I showed that remerge can be internal as well as external. These phenomena may be exemplified by *wh*-movement and Right Node Raising, respectively. Internal remerge is commonly accepted, but external remerge is not. However, both possibilities follow from the core system; if one or both are to be excluded, this has to be stipulated explicitly. It is straightforward how to exclude remerge in general, but conditions to the effect that only one type (in particular, external remerge) is

[22] The situation for external remerge leads to comparable difficulties (I will not discuss this in detail). Moreover, an additional problem arises: the shared constituent must be first-merged in the cycle in which it is to be phonologically elided (for instance, the first conjunct in a RNR construction); however, what will be the first conjunct is only decided higher up in the structure, so this is a form of look-ahead.

[23] Bachrach & Katzir (2006) claim that their analysis of RNR (which is also in put terms of remerge) involves spell-out in cycles. However, they seem to overlook that their notion of complete dominance implies that the linearization procedure is able to 'see' the complete structure (which is necessary if some node Y has parents in different phases).

excluded are far from trivial. I argued against such stipulations, especially since there are sensible interpretations of (external) remerge. Furthermore, I showed that both internal and external remerge can be represented in terms of multidominance. (A set notation is not fit for remerge in general.) In passing, I noted that the concept of 'interarboreal movement' is an artifact of the notation. The unification of internal and external remerge makes it particularly clear that there is no need for a copying mechanism, for traces or for (movement) chains. Thus, the syntactic apparatus can be kept to a minimum.

Nevertheless, it is evident that the possibility of remerge creates difficulties for the linearization of syntactic objects, which is required at the PF interface. Moreover, I showed that there seem to be contradictory linearization demands for internal and external remerge. However, I proposed a solution in the form of a conditioned linearization algorithm, which makes use of the different structural configurations created by the two types of remerge.

Furthermore, I emphasized that a syntactic tree is just a representation of the outcome of a sequence of mergers. What Merge does, is to establish primitive relations between syntactic objects. These relations are inclusion (dominance), and asymmetry (aboutness). At the PF interface, the syntactic/semantic asymmetry between siblings is mapped onto precedence. The result of a derivation is a list of primitive relations, which is equivalent to a list of basic triads. A complex tree representation (possibly involving multidominance) can be established on the basis of such a list rather straightforwardly. I also showed what the proposed tree scanning algorithm does in terms of a list of primitive relations, and I argued that the computational load of this procedure is acceptable. Finally, I argued that the linearization procedure is not likely to operate in cycles.

References

Armstrong, D. M. 1986. In Defense of Structural Universals. *Australasian Journal of Philosophy* 64, 85–88.

Bachrach, Asaf & Roni Katzir. 2006. Right-Node Raising and Delayed Spellout. Ms. MIT.

Blevins, James. 1990. *Syntactic Complexity: Evidence for Discontinuity and Multidomination*. Doctoral dissertation, University of Massachusetts.

Bobaljik, Jonathan. 1995. In Terms of Merge: Copy and Head Movement. *MIT Working Papers in Linguistics* 27, 41–64.

Bobaljik, Jonathan & Samuel Brown. 1997. Interarboreal Operations: Head Movement and the Extension Requirement. *Linguistic Inquiry* 28, 345–356.

Carstens, Vicky. 2003. Rethinking Complementizer Agreement: Agree with a Case-checked Goal. *Linguistic Inquiry* 34: 393–412.

Chomsky, Noam. 1981. *Lectures on Government & Binding*. Dordrecht: Foris.

Chomsky, Noam. 1995. *The Minimalist Program*. MIT Press, Cambridge, Mass.

Chomsky, Noam. 2004. On phases. Ms. MIT.

Citko, Barbara. 2005. On the Nature of Merge: External Merge, Internal Merge, and Parallel Merge. Ms. Brandeis University.

Collins, Chris. 2002. Eliminating Labels. In *Derivation and Explanation in the Minimalist Program*, ed. Samuel Epstein & Daniel Seely, 42–64. Oxford: Blackwell.

Cormen, Thomas et al. 1990. *Introduction to Algorithms*. Cambridge, Mass.: MIT Press.

Culicover, Peter & Ray Jackendoff. 1997. Semantic subordination despite syntactic coordination. *Linguistic Inquiry* 28, 195–217.

Di Sciullo, Anna Maria. 2000. Asymmetries: Consequences for Morphological Configurations and Paradigms. *Acta Linguistica Hungarica* 47, 81–101.

Epstein, Samuel. 1999. Un-Principled Syntax and the Derivation of Syntactic Relations. In *Working minimalism*, ed. Samuel Epstein & Norbert Hornstein, 317–345. Cambridge, Mass.: MIT Press.
Fiengo, Robert. 1974. *Semantic Conditions on Surface Structure*. Doctoral dissertation, MIT.
Forrest, Peter. 1986. Neither Magic nor Mereology: A Reply to Lewis. *Australasian Journal of Philosophy* 64, 89–91.
Fox, Danny & David Pesetsky. 2005. Cyclic Linearization of Syntactic Structure. *Theoretical Linguistics* 31, 1–45.
Frampton, John. 2004. Copies, Traces, Occurrences, and All That: Evidence from Bulgarian Multiple *Wh*-Phenomena. Ms. Northeastern University.
Fukui, Naoki. & Yuji Takano. 1998. Symmetry in Syntax: Merge and Demerge. *Journal of East Asian Linguistics* 7, 27–86.
Gärtner, Hans-Martin. 2002. *Generalized Transformations and Beyond: Reflections on Minimalist Syntax*. Berlin: Akademie Verlag.
Goodall, Grant. 1987. *Parallel Structures in Syntax: Coordination, Causatives and Restructuring.* Cambridge: Cambridge University Press.
Goodman, Nelson. 1961. About. *Mind* 70, 1–24.
Goodman, Nicolas. 1986. Intensions, Church's Thesis, and the Formalization of Mathematics. *Notre Dame Journal of Formal Logic* 28, 473–489.
Grootveld, Marjan. 1994. *Parsing Coordination Generatively.* Doctoral dissertation, Leiden University.
Guimarães, Maximiliano. 2004. *Derivation and Representation of Syntactic Amalgams.* Doctoral dissertation, University of Maryland.
Hartmann, Katharina. 2000. *Right Node Raising and Gapping: Interface Conditions on Prosodic Deletion.* Amsterdam: John Benjamins.
Heijden, Emmeke van der. 1999. *Tussen nevenschikking en onderschikking*. Doctoral dissertation, Catholic University of Nijmegen.
Huybregts, Rini. & Henk van Riemsdijk. 1985. Parasitic Gaps and ATB. *Proceedings of NELS* 15, 168-187.
Johnson, Kyle. 2007. LCA + alignment = RNR. Paper presented at the workshop on Coordination, Subordination and Ellipsis, Tübingen, 7/8 June 2007.
Karttunen, Lauri & Martin Kay. 1986. Structure Sharing with Binary Trees. In *A Compilation of Papers on Unification-based Grammar Formalisms, Part II*, ed. Stuart Shieber et al., 5–16. CSLI-86–84 Report, Stanford.
Kayne, Richard. 1984. *Connectedness and Binary Branching*. Dordrecht: Foris.
Kayne, Richard. 1994. *The Antisymmetry of Syntax*. Cambridge, Mass.: MIT Press.
Kluck, Marlies. To appear. The perspective of external remerge on Right Node Raising. *Proceedings of CamLing 2007*.
Koster, Jan. 1999. De primaire structuur. *TABU* 29: 131–140.
Koster, Jan. 2003. All Languages are Tense Second. In *Germania et Alia: A Linguistic Webschrift for Hans den Besten*, ed. Jan Koster & Henk van Riemsdijk. http://www.let.rug.nl/~koster/DenBesten/contents.htm.
Koster, Jan. 2007. Structure Preservingness, Internal Merge, and the Strict Locality of Triads. In *Phrasal and Clausal Architecture: Syntactic derivation and interpretation* [In honor of Joseph E. Emonds], ed. Simin Karimi, Vida Samiian & Wendy K. Wilkins, 188–205. Amsterdam: John Benjamins.
Kremers, Joost. 2007. Recursive linearisation. Ms. J.W. Goethe University, Frankfurt.
Kuratowski, Kazimierz. 1921. Sur la notion de l'ordre dans la théorie des ensembles. *Fundamenta Mathematicae* 2, 161–171. Reprinted in Kazimierz Kuratowski, *Selected Papers*, 1988. Warszawa: Polish Scientific Publishers.
Langendoen, Terence. 2003. Merge. In *Formal Approaches to Function in Grammar. In Honor of Eloise Jelinek*, ed. Andrew Carnie et al., 307–318. Amsterdam: John Benjamins.

McCawley, James. 1968. Concerning the Base Component of a Transformational Grammar. *Foundations of Language* 4, 243–269.
McCawley, James. 1982. Parentheticals and Discontinuous Constituent Structure. *Linguistic Inquiry* 13, 91–106.
Moltmann, Friederike. 1992. *Coordination and Comparatives.* Doctoral dissertation, MIT.
Mu'adz, Husni. 1991. *Coordinate Structures: A Planar Representation.* Doctoral dissertation, University of Arizona.
Neijt, Anneke. 1979. *Gapping: A Contribution to Sentence Grammar.* Dordrecht: Foris.
Nilsen, Øystein. 2005. Some notes on cyclic linearization. *Theoretical Linguistics* 31, 173–183.
Nunes, Jairo. 2001. Sideward Movement. *Linguistic Inquiry* 32, 303–344.
Ojeda, Almerindo. 1987. Discontinuity, Multidominance, and Unbounded Dependency in Generalized Phrase Structure Grammar: Some Preliminaries. In *Syntax & Semantics, Vol. 20, Discontinuous Constituency*, ed. Geoffrey Huck & Almerindo Ojeda, 257–282. Orlando: Academic Press.
Postal, Paul. 1998. *Three Investigations of Extraction*. Cambridge, Mass.: MIT Press.
Quine, Willard Van. 1945. On Ordered Pairs. *The Journal of Symbolic Logic* 10, 95–96.
Reinhart, Tanya. 1982. *Pragmatics and linguistics: An analysis of sentence topics*. Bloomington, Indiana: Indiana University Linguistics Club.
Riemsdijk, Henk van. 1998. Trees and Scions – Science and Trees. In *Chomsky 70th Birthday Celebration Fest-Web-Page*. http://cognet.mit.edu/Books/celebration/essays/riemsdyk.html.
Riemsdijk, Henk van. 2006. Grafts Follow from Merge. In *Phases of Interpretation*, ed. M. Frascarelli, 17-44. Berlin: Mouton de Gruyter.
Ross, John. 1967. *Constraints on Variables in Syntax*. Doctoral dissertation, MIT. Printed as *Infinite syntax!* 1986. Norwood, New Jersey: ABLEX.
Sag, Ivan & Janet Fodor. 1994. Extraction Without Traces. *West Coast Conference on Formal Linguistics* 13, 365–384. Stanford: CSLI Publications.
Sampson, Geoffrey. 1975. The Single Mother Condition. *Journal of Linguistics* 11, 1–11.
Schneider, Hubert. 1977. Some Remarks Concerning a Definition of Ordered Pairs. *The American Mathematical Monthly* 84, 636–638.
Sider, Theodore. 1996. Naturalness and Arbitrariness. *Philosophical Studies* 81, 283–301.
Starke, Michal. 2001. *Move Dissolves into Merge: a Theory of Locality*. Doctoral Dissertation, University of Geneva.
Velde, John te. 1997. Deriving Conjoined XPs: A Minimal Deletion Approach. In *German: Syntactic Problems – Problematic Syntax*, ed. Werner Abraham & Elly van Gelderen, 231–259. Tübingen: Max Niemeyer.
Vries, Gertrud de. 1992. *On Coordination and Ellipsis*. Doctoral dissertation, University of Tilburg.
Vries, Mark de. 2005a. Coordination and Syntactic Hierarchy. *Studia Linguistica* 59, 83–105.
Vries, Mark de. 2005b. Ellipsis in nevenschikking: voorwaarts deleren, maar achterwaarts delen. *TABU* 34, 13–46.
Vries, Mark de. 2005c. Merge: Properties and Boundary Conditions. *Linguistics in the Netherlands* 22, 219–230.
Wiener, Norbert. 1914. A Simplification of the Logic of Relations. *Proceedings of the Cambridge Philosophical Society* 17, 387–390. Reprinted in *From Frege to Gödel: A Source Book in Mathematical Logic, 1879–1931*, 1967, ed. Jean van Heijenoort. Cambridge, Mass.: Harvard University Press.
Wilder, Chris. 1997. Some Properties of Ellipsis in Coordination. In *Studies on Universal Grammar and Typological Variation*, ed. Artemis Alexiadou & Alan Hall, 59–107. Amsterdam: John Benjamins.

Wilder, Chris. 1999. Right Node Raising and the LCA. In *Proceedings of the 18th West Coast Conference on Formal Linguistics (WCCFL 18)*, ed. Sonya Bird et al., 586–598. Somerville, Mass.: Cascadilla Press.
Williams, Edwin. 1978. Across-The-Board Rule Application. *Linguistic Inquiry* 9: 31–43.
Yasui, Miyoko. 2002. A Graph-Theoretic Reanalysis of Bare Phrase Structure Theory and its Implications on Parametric Variation. Paper presented at Linguistics and Phonetics 2002, Meikai University, Urayasu.
Yasui, Miyoko. 2004. Syntactic Structure without Projection Labels. Ms. Dokkyo University.
Zhang, Niina. 2004. Move is Remerge. *Language and Linguistics* 5, 189–209.
Zwart, Jan-Wouter. 1994. Dutch is Head Initial. *The Linguistic Review* 11, 377–406.
Zwart, Jan-Wouter. 2004. Een dynamische structuur van de Nederlandse zin. Deel 1 en 2'. *TABU* 33, 55–71 and 151–172.
Zwart, Jan-Wouter. 2006. Local Agreement. In *Agreement Systems*, ed. Cedric Boeckx, 317–339. Amsterdam: John Benjamins.

Author's address:
Mark de Vries
Dept. of Linguistics (ATW)
University of Groningen
P.O. Box 716
9700 AS GRONINGEN
The Netherlands

e-mail: mark.de.vries@rug.nl