On Multidominance and Linearization

Mark de Vries

This article centers around two questions: What is the relation between movement and structure sharing, and how can complex syntactic structures be linearized? It is shown that regular movement involves *internal remerge*, and sharing or 'sideward movement' *external remerge*. Without *ad hoc* restrictions on the input, both options follow from Merge. They can be represented in terms of multidominance. Although more structural freedom ensues than standardly thought, the grammar is not completely unconstrained: Arguably, proliferation of roots is prohibited. Furthermore, it is explained why external remerge has somewhat different consequences than internal remerge. For instance, apparent non-local behavior is attested. At the PF interface, the linearization of structures involving remerge is non-trivial. A central problem is identified, apart from the general issue why remerged material is only pronounced once: There are seemingly contradictory linearization demands for internal and external remerge. This can be resolved by taking into account the different structural configurations. It is argued that the linearization is a PF procedure involving a recursive structure scanning algorithm that makes use of the inherent asymmetry between sister nodes imposed by the operation of Merge.

Keywords: linearization; movement; multidominance; PF interface; (re-) merge

1. Introduction and Overview

Displacement is one of the central tenets in generative grammar. The underlying idea is that a word or phrase may be involved in more than one relationship; therefore, it can be associated with a sentence position where it does not surface. A simple example in English is *wh*-movement, such as illustrated in (1):

I thank the anonymous referees for useful comments and questions. Special thanks also to Jan Koster, Leonie Bosveld, Marlies Kluck, Herman Heringa, Eric Hoekstra, Jan-Wouter Zwart, Anneke Neijt, Anna Maria Di Sciullo, Hans Broekhuis, Henk van Riemsdijk, Hans-Martin Gärtner, Eric Reuland, and Lisa Cheng. Of course, all remaining shortcomings are my own. This research was financially supported by the Netherlands Organisation for Scientific Research (NWO).

biolinguistics
© The Author (2009)

ISSN 1450–3417

(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*. This phrase is categorically and semantically selected by the verb *purchase*. In (1b), the preposed object *which violin* is thought to be related to the regular direct object position next to the main verb as well, here indicated by an underscore. How does the grammar make sure that the object is pronounced in the higher, operator-related position (leftmost), and not in the lower, thematic position (rightmost)? A fairly standard approach in generative grammar has been the assumption that movement is hierarchically directional, and that a moved phrase leaves an unpronounced trace in the original lower position. In current minimalist theories, specialized traces no longer exist (this follows from the Inclusiveness condition proposed by Chomsky 1995: 225). From the perspective of a bottom-up derivation, it seems that we must make sure that the first occurrence of the relevant phrase (here, *which violin*) remains phonologically silent if, after movement, there will be a second, higher occurrence of it. Clearly, then, the linearization of a sentence structure is a non-trivial process taking place at the interface between syntax and phonology. This article is an attempt to explicate that process, and its preconditions.

An interesting complication is that there appear to be constructions that essentially show the opposite pattern, though not exactly in a mirror fashion. A relevant example is the so-called Right Node Raising (RNR) construction. In (2), *this beautiful Stradivarius* is the object of *admired* as well as *bought*, but here only the rightmost occurrence is spelled out, contrary to the situation in (1b).

(2) The boy only admired ___, but the girl actually bought *this beautiful Stradivarius*.

Though I do not think that there is rightward or lowering movement, there are reasons to believe that phrases can be structurally *shared*, which could be represented by a multidominance configuration (this will be explained below). The questions we then face are the following:

(Q1) How are sharing configurations derived, and what is the theoretical relationship with movement?

(Q2) When and how does the linearization procedure operate, and how does it distinguish between the two different construction types illustrated by (1b) and (2), respectively?

In section 2, I argue that a freely applicable operation Merge gives rise to the possibility of both *internal remerge* and *external remerge*. The concept of movement corresponds to the first, and that of sharing to the second. I should mention right away that this article is not about the correct analysis of RNR, *wh*-movement, or any other particular construction still to be mentioned. Rather, I intend to explore the theoretical consequences of remerge. References to particular

analyses are used for concreteness' sake, and serve as illustrations, mainly.

Since the syntactic configurations arising by applying the two types of remerge are different, the linearization procedure can be made sensitive to it. This is the subject of section 3, which presents a solution to the seemingly contradictory linearization demands briefly introduced here. It is claimed that linearization involves the scanning (traversal) of a full sentence structure. Various complicated construction types are examined. Section 4 presents a more detailed graph scanning algorithm, and discusses the computational load of such a procedure, taking into account the difference between representations and the actual theoretical assumptions. Finally, section 5 is the conclusion.

2. Internal and External Remerge

2.1. The Operation Merge: Input and Output

The input for Merge, which I assume to be binary (following standard assumptions dating back to Kayne 1984), is restricted to objects recognizable by syntax, that is, words and phrases — or rather the features associated with these. Nevertheless, judging from general minimalist practice since Chomsky (1995), the *selection* of these objects must be free with respect to their location or history. There are three possibilities, two of which are logically necessary if Merge is the only structure-building device. First, input objects for Merge can be selected from the lexicon or 'numeration'. Of course, syntax would be idle without subject matter. Second, the result of a previous instance of Merge can be selected as the input for a subsequent instance of Merge. This corresponds to the general hierarchical aspect of syntax. Without the recursive application of Merge, objects more complex than two words could never be derived. Non-trivial objects are created in the syntactic workspace. It is not only the active structure itself that is complex after first Merge: Auxiliary structures are also necessary. For example, subjects and adverbial phrases are often complex (notice that even a simple noun phrase like *the man* counts as such, as it consists of more than one element). If they are to be attached to the main projection line, they must have been derived already in an auxiliary derivation.

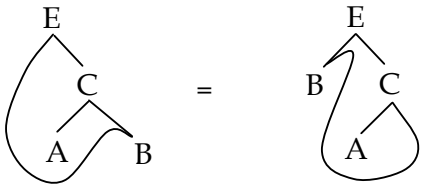
The third possibility is fairly standard as well, though not undisputed. Let us assume that there is such a thing as displacement, as indicated in the introduction. Displacement from a derivational perspective implies that a constituent ('term') of a derived structure is accessible as a possible input object for another instance of Merge. It follows that a syntactic object can be merged more than once. In this way, we account for the fact that syntactic objects can be involved in more than one relationship, associated with different positions in the structure. Here, it is presupposed that grammatical relationships are a direct or indirect function of Merge, and hence of structure. This is a central insight of generative grammar, and I will not question it.

To sum up, three differently situated kinds of objects may serve as input for Merge: (i) lexical items, (ii) complex items that are the result of previous instances of Merge, (iii) terms of complex items. The option in (iii) normally

corresponds to what is often called Move (Chomsky 1995). However, it is important to see that there is only one basic operation, Merge. Depending on the input, the result may be Move. If Move involves the creation of traces or copies with special properties, it would constitute a separate, complex operation. However, according to minimalist reasoning, this cannot be *a priori* assumed. In recent work, Chomsky refers to iii) as *internal merge*, as opposed to *external merge* for (i) and (ii), stressing that the possibility of movement simply follows from Merge (Chomsky 2001a). One could also say that the distinction is between *(first-time) merge* and *remerge* (that is, Merge again). The first, *merge*, is inevitably external. But is *remerge* always internal? Standardly, this is tacitly assumed. However, it does not in any way follow from the definition of Merge, or from the boundary conditions mentioned so far. This will become clear in a moment.

The essence of Merge is that it is structure-building. It combines units into a larger unit, which then constitutes a new root. In accordance with the two usual boundary conditions, it combines two distinct syntactic objects (say, A and B) into a new, larger unit (C), which, by definition, is then also a syntactic object. Let us notate this as Merge (A, B) → C, which is an operation resulting in the possible representation [$_C$ A B]. If we go on merging C with an external D (lexical or complex), we create another syntactic object, call it E: Merge (D, C) → E, resulting in [$_E$ D [$_C$ A B]]. If instead we merge a term of C, say B, with the root C again, we create a movement configuration by internal remerge: Merge (B, C) → E, giving [$_E$ B [$_C$ A B]], where B has now two sisters (that is, Merge-mates), namely both A and C. We are used to calling the lowest B a copy, but this is misleading: Nothing in the syntactic system distinguishes the two Bs in the representation (unless further, complicating assumptions are made). In fact, there is no second B to begin with. There is just one B that is involved in two relationships created by Merge. The two Bs are an artifact of the representation. A less misleading way of representing the result of the two mergers under discussion is the multidominance representation in (3), although it has the disadvantage of being graphically a little awkward. Notice that we can picture B in its first-merge position, in its Spell-Out position, or in fact anywhere else on the paper:

(3) Merge (A, B) → C
Merge (B, C) → E



See also Epstein *et al.* (1998), Starke (2001), Gärtner (2002), Zhang (2004), and Frampton (2004), among others, for further arguments against the copying view of displacement.[1] For earlier discussion of similar ideas, see Sampson (1975), Karlgren (1976), McCawley (1982), Peters & Richie (1982), Engdahl (1986), Huck & Ojeda (1987), Blevins (1990). What should be clear is that the assumption of

[1] From a completely different perspective, Karttunen & Kay (1985) warn that the amount of computational effort that goes into producing copies is much greater than the cost of 'unification' (that is, multidominance) when a graph is being parsed. For this reason, they advocate structure sharing.

copies would require theoretical machinery in addition to the operation Merge *per se*. A different matter is how the phonological interface interprets the result in (3); this will be discussed in detail in section 3.

Movement, as we saw, involves *remerge*, that is, a syntactic 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 R from which α has been selected, this instance of remerge can be called *internal*. However, as announced before, this does not exhaust the possibilities: α can in principle be remerged with an independent syntactic object, that is, an object that is not R and not embedded in R. This is what I will call *external remerge*; see (4).[2]

(4) 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 root β not included in R).

Crucially, there is just one operation Merge; labels such as *internal remerge* are just names for the different situations caused by selecting different input objects. This is expressed in (5):

(5) Merge (α, β) → γ constitutes
   a. *first-time merge* iff α and β are independent roots before merger;
   b. *internal remerge* iff β is a root and α is included in β (or the other way around) before merger;
   c. *external remerge* iff β is included in some root δ, and α is an independent root (or the other way around) before merger.

Notice that heads introduced from lexicon or numeration are (trivial) roots before they are merged. For discussion concerning the strict cycle, I refer to section 2.3..

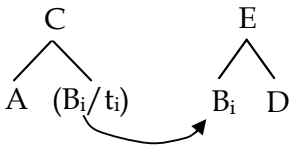
Although external remerge leads to unconventional structures (see further below), I must be stressed that the possibility of this operation simply follows from the combination of two independently motivated options: The selection of external material as input for Merge (needed for the introduction of lexical material), and the selection of terms (needed for regular movement); see also de Vries (2005c) and van Riemsdijk (2006a). It is of course possible to impose stricter boundary conditions on the input for Merge. For instance, the input could be restricted to roots. The consequence of this would be that remerge is excluded altogether (including regular movement). This point of view is defended in

[2] As far as I know, Barbara Citko, Henk van Riemsdijk, and I myself first published basically equivalent ideas around 2005, independently of each other, and with somewhat differing terminology. In fact, it was predated by a remark in Wilder (1999), and of course inspired by earlier work on interarboreal movement, among others (see further below in the main text). It is perhaps worth mentioning that Chomsky (2007: 8, fn. 10) does not seem to agree: "[External remerge] requires new operations and conditions on what counts as a copy". Further explanation is lacking, and frankly, I fail to see why this would be so. Moreover, the objection is invalid from the present perspective, since a copying mechanism was rejected to begin with.

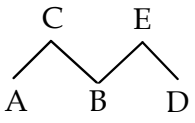
Koster (2007), among others.[3] The grammar would then be more restricted, but at the cost of an additional rule. If the familiar internal remerge is to be allowed, but the unorthodox external remerge to be excluded, more specific additional conditions must be formulated. However, it may be interesting to put off such stipulations, and allow for remerge in general. Here, I will follow this track, and explore some of the consequences.

Several possible interpretations of what can now be recognized as external remerge have been proposed in the literature. These include 'interarboreal movement' (Bobaljik 1995, Bobaljik & Brown 1997), 'sideward movement' (Nunes 2001), 'multidominance/multidomination/multiple dominance' (McCawley 1982, Ojeda 1987, Blevins 1990, Wilder 1999, Chen–Main 2006, Johnson 2007, Bachrach & Katzir 2009), 'sharing' (Guimarães 2004, Chung 2004, de Vries 2005b, Gracanin–Yuksek 2007), 'grafting' (van Riemsdijk 1998, 2006a), and 'parallel merge' (Citko 2005). Furthermore, external remerge is allowed in some way or another in many theories involving 'parallel structures' (Williams 1978, Goodall 1987, Mu'adz 1991, G. de Vries 1992, Moltmann 1992, Grootveld 1994, te Velde 1997). See also Carnie (2008) for a brief overview. I cannot do justice to all these proposals, but the two central ideas that are relevant here are pictured in (6). Notice that (6a) equals (6a'), and (6b) equals (6b'); apparent differences are only due to the position of the independent two-legged mini-structures on the paper:

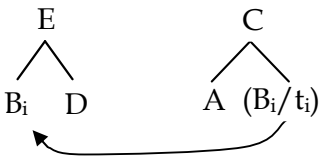
(6) a.



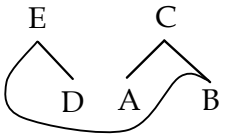
b.



a'.



b'.



In (6a/a'), B is moved to an independent structure. Let us provisionally call this iMove (short for interstructural movement). This iMove is different from traditional (rightward or leftward) movement, which involves movement to a position within or at the top of the *same* structure. In (6b/b'), B is shared between two structures. Let us call this mDom (short for a hydraic — that is, multi-rooted — multiple dominance configuration), which, like internal remerge as in (3), involves giving up the 'single mother condition' used in previous frameworks

[3] Koster (2007) argues against 'internal [re]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, for example, Sag & Fodor (1994). Another take on the issue is put forward by Blevins (1990), who eliminates movement by treating order as completely independent from hierarchical structure; this is inspired by earlier work by Sampson (1975) and McCawley (1968, 1982).

(see Sampson 1975); in a derivational framework, it also involves giving up the 'single root condition' — at least during the derivation (see further section 2.3). However, if structures are derived by Merge, *all* representations in (6) are derived by the following two applications of Merge:

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

In (7a), B is merged with A, which gives C. In (7b), B is remerged with D, which gives E. Since D is not related to C (the root), the step in (7b) is an instance of external remerge. Thus, the perhaps surprising conclusion must be that it is only the notation that suggests a difference between iMove and mDom, captured as *external remerge:* iMove = mDom.

Without further assumptions (such as special properties of copies/traces and chains, which we must reject *a priori* until strong independent evidence to the contrary comes up, *pace* Nunes 2001 and others), iMove is actually equivalent to mDom. The representations in (6) are just that: More or less successful representations of certain theoretical concepts. What is 'real' is that Merge creates basic relationships between syntactic objects: Grammatical inclusion and grammatical sisterhood (see section 4 for further discussion). A graph that represents such relationships has no independent theoretical status. See also de Vries (2009b) on the issue of notation in syntax, including an unorthodox proposal. Furthermore, I would like to stress that multidominance is independent of *multidimensionality* (e.g., '3D grammar'), despite some suggestive descriptions in the literature. An additional syntactic dimension, in my view, would imply the assumption of an additional basic relationship (next to dominance or sisterhood); see also Grootveld (1994).[4]

In (7), there is only one B, and this B is engaged in two basic 'triads', if I may borrow an expression from Koster (2007). A triad is the minimum amount of structure, equivalent to what is created by one instance of Merge. Thus, Merge (α, β) → γ relates α, β and γ such that α and β are directly included in γ, and α is the grammatical sister of β. The advantage of using multidominance graphs as in (6) is that they represent the fact that some node (here, B) is involved in a double set of basic relationships, without suggesting that this node itself is magically multiplied. The mDom notation, therefore, can be used to represent remerge in general, and I will stick to it in the remainder of this article.

2.2. *Potential Examples of External Remerge*

In section 2.3, I will address the status of the strict cycle and some other theoretical issues, but first let me provide some concrete examples of sentence

[4] Such proposals exist for both parenthesis and coordination (see also de Vries 2005a, 2007, 2009b for discussion and further references, some of which are mentioned in the main text). Strict definitions aside, it seems clear that all four combinations of [±3D] and [±mDom] occur: There are 'parallel structures' with and without 'sharing', and there may be remerge in regular hypotactic configurations as well. See the next subsection for some examples of the <+, +> pattern.

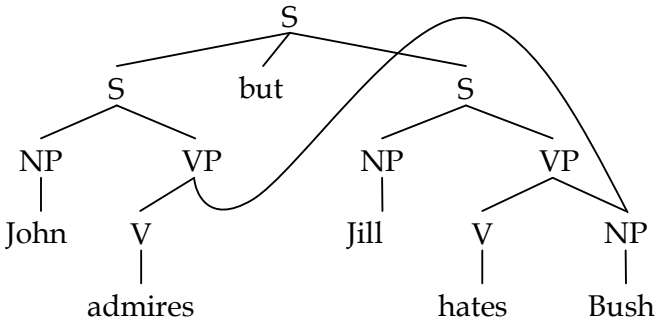
structures that may involve external remerge.

A by now almost classic case is RNR (or backward conjunction reduction). A simple example is provided in (8):

(8) John admires ___, but Jill hates Bush.

The implied object in the first conjunct is *Bush*. McCawley (1982) proposed that this construction can be analyzed by allowing a constituent to be shared between two conjuncts, as is depicted in (9) — my example, with a simplified sentence structure for expository purposes. Here, the object *Bush* is dominated by both verb phrases:

(9)



Although it is has not remained uncontested (see Postal 1998, Sabbagh 2007, Ha 2008a, 2008b), the idea of applying multidominance to RNR has been picked up and defended by several authors, for instance, Ojeda (1987), G. de Vries (1992), Wilder (1999, 2008), Chung (2004), de Vries (2005b), Chen–Main (2006), Johnson (2007), Kluck (2007, 2009), Bachrach & Katzir (2009), and Kluck & de Vries (to appear). Even though it is cast in different frameworks and stages of general syntactic theory, the basic idea is still the same. From the present perspective, we would say that the derivation of (9) involves merger of the NP *Bush* with one of the verbs, and then it remerges with the other verb.[5] Temporarily, this leads to a doubly-rooted structure, but since the two conjuncts are united at the top, the problem is resolved. (I will return to this.)

The reason for treating RNR in this special way is that it behaves differently from forward ellipsis/deletion, and also from regular movement and extraposition. For instance, RNR is apparently insensitive to island conditions (see Neijt 1979 and Hartmann 2000, among others; see also below), and it is immune to the Head condition on remnants (Fiengo 1974, Wilder 1997). Both properties fall out naturally from a multidominance approach. Trivially, since there is no ellipsis, there are no remnants, so the head condition does not apply, as

[5] A concern for a theory in which an argument can be shared is that the relevant DP is assigned a theta-role twice (or more). It is conceivable that this is only allowed if these theta-roles are identical. Indeed, it is hard to imagine acceptable instances of RNR involving semantically different types of arguments. Thus, the matching effect induced by structure sharing may in fact serve as an explanation of certain parallelism requirements in reduced coordinated clauses. Notice that the situation is different in amalgams (see below); here, what is shared functions as a predicate in the interrupting clause, so the issue of a double theta-role does not arise (Kluck, in progress).

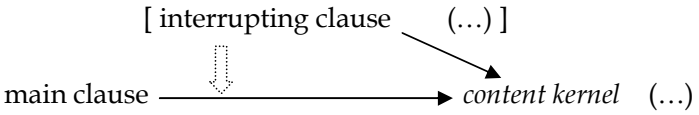
required. Furthermore, no matter how deeply embedded the shared constituent is (here, the NP *Bush*), it is locally related to each sister (here, the two verbs). That is, the multidominance connection creates a kind of bypass (see also section 2.4). For more discussion concerning RNR *per se*, see Kluck & de Vries (to appear) and the references mentioned.

Other constructions that qualify for external remerge are *wh-amalgams* and *cleft-amalgams*, as discussed in Guimarães (2004) and Kluck (2008), based on earlier work in Lakoff (1974), van Riemsdijk (1998), and Tsubomoto & Whitman (2000) — *pace* Zwart (2006b) and Grosu (2006). These are illustrated in (10a) and (10b), respectively.

(10) a. Jack gave [you will never guess which *girl*] a flower.
b. Jack gave [I think it was *his girlfriend*] a flower.

Here, the interrupting clause between brackets gives rise to a bracketing paradox, since the *content kernel* in italics is also part of the main clause. A multidominance solution to this problem is informally sketched in (11):

(11)



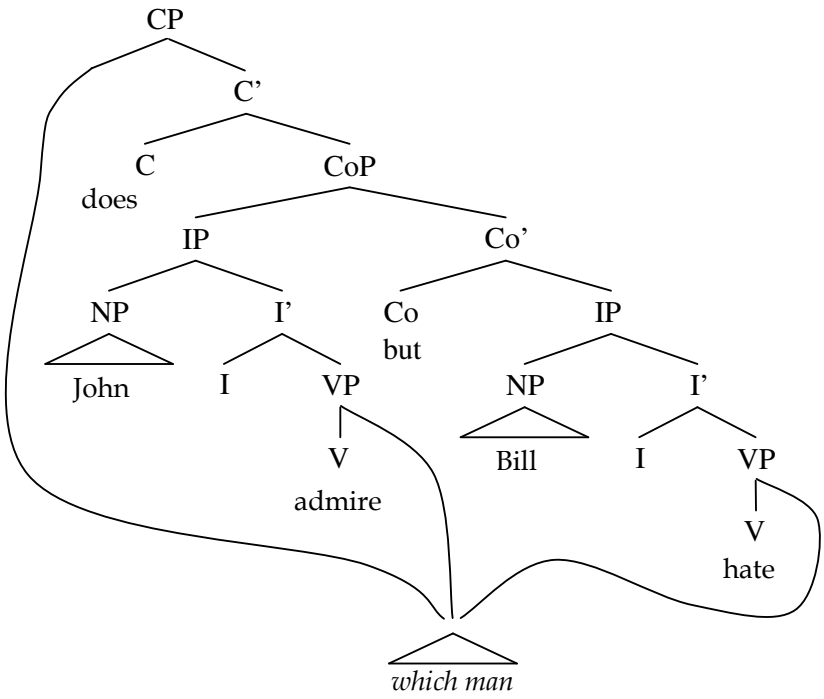
The content kernel is dominated by a projection of the main clause as well as the interrupting clause. The latter is inserted as a parenthetical in the main clause (Kluck, in progress, *contra* Guimarães 2004; see also de Vries 2009b). The details need not concern us here; what is relevant is that the shared constituent needs to be externally remerged.

Another construction that has been argued to involve sharing is Across-the-Board (ATB) movement; see Williams (1978), Goodall (1987), Citko (2005), Mayr & Schmitt (2009), among others. A standard example is (12):

(12) *Which man* does John admire ___ but Bill hate ___?

The idea is that prior to *wh*-movement the relevant constituent (here, the object *which man*) is shared between positions within two or more conjoined clauses (IPs); see (13):

(13)



This structure is derived by externally remerging the object from one VP to the other; after that, both conjuncts are completed and joined by means of a coordination phrase;[6] finally, the CP level is added, and regular *wh*-movement takes place. Thus, the ATB-construction combines external and internal remerge. In section 3 it is discussed how it must be linearized.

Let me list some further, interesting proposals that involve externally remerged material (for the record, I am not personally committed to all of these). In chronological order:

- van Riemsdijk (1998, 2006b) on transparent free relatives, where the content kernel (the predicate) of the TFR is shared with the matrix (he also suggests a similar approach to internally headed relative clauses;
- Nunes (2001, 2004) on parasitic gap constructions, where the *wh*-constituent is 'sideward moved' before fronting;
- van Riemsdijk (2001a) on *wh*-prefixes, where the *wh*-word is shared between the matrix and the 'prefix' (for instance, "God knows who…");
- van Riemsdijk (2001b) on bracketing paradoxes as in *a far from simple matter*, where the adjective is part of two different trees;
- van Riemsdijk (2006b) on regular free relatives, where there is sharing of the *wh*-operator between the matrix and the subordinate clause (the purpose of this is to explain Case matching effects).
- Henderson (2007) on relative clauses, where there is sideward movement of the head NP between the relative clause and the matrix;

[6] For discussion and references concerning coordination *per se*, see de Vries (2005a).

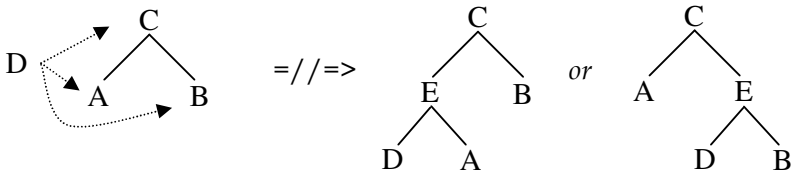
- Gracanin–Yuksek (2007) on coordinated *wh*-constructions, where there is 'bulk sharing' after the *wh*-constituents;
- Meinunger (2008) on bracketing paradoxes in certain complex numerals;
- Heringa (2009, in progress) on appositional constructions, where the appositional core is shared between a parenthetical position and a position in the matrix;

Whether each individual analysis of a particular phenomenon just mentioned will eventually be embraced or discarded does not matter for the purpose of this article. The point is that there is a by now substantial body of literature on structures involving external remerge. This in itself justifies a closer look at the formal properties of sharing, and the problem of linearization in comparison with regular movement.

2.3. The Strict Cycle

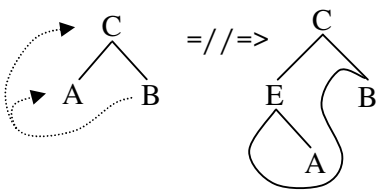
The possibility of *remerge* raises questions about the course derivations can take. In this respect, consider the so-called *extension condition*, also known as *strict cyclicity* (Chomsky 1995: 190, 327). Since Merge is structure-building and not structure-changing, counter-cyclic *merge* or *remerge* is simply impossible. Basically, Merge (X, Y) combines X and Y but leaves the internal structure of X and Y intact. This is worked out in some more detail in (14) and (15), for merge and remerge, respectively. In each case, the projection E is created, but E is not the new root. Instead, E is inserted as the daughter of C, and the original direct inclusion relationship between C and A in (14a) and (15a), and the one between C and B in (14b) and (15b) is destroyed. In each example, the original existence of [$_C$ A B] is the result of a previous instance of Merge.

(14) a. (i) Merge (D, [$_C$ A B]) −//→ [$_C$ [$_E$ D A] B]
(ii) [$_C$ A B] and Merge (D, A) −//→ [$_C$ [$_E$ D A] B]
b. (i) Merge (D, [$_C$ A B]) −//→ [$_C$ A [$_E$ D B]]
(ii) [$_C$ A B] and Merge (D, B) −//→ [$_C$ A [$_E$ D B]]

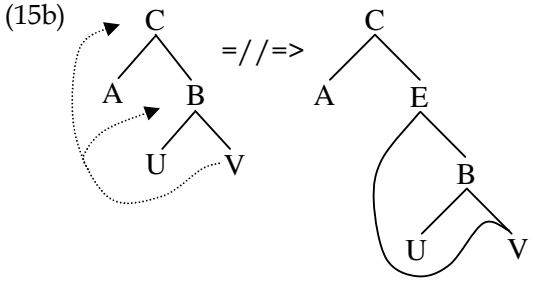


(15) a. (i) Merge (B, [$_C$ A B]) −//→ [$_C$ [$_E$ B A] B] (mDom of B)
(ii) [$_C$ A B] and Merge (B, A) −//→ [$_C$ [$_E$ B A] B] (mDom of B)
b. (i) Merge (V, [$_C$ A [$_B$ U V]]) −//→ [$_C$ A [$_E$ V [$_B$ U V]] (mDom of V)
(ii) [$_C$ A [$_B$ U V]] and Merge (V, B) −//→ [$_C$ A [$_E$ V [$_B$ U V]](mDom of V)

(15a)



(15b)



Clearly, if this were possible, it would be an undesirable complication of the theory. A similar reasoning can be found in Chomsky (2005), who introduces the *no-tampering condition*. The no-tampering condition can be considered a derived consequence of the system. It need not be an independent principle of grammar, since tampering is simply not what Merge does, at least not from the most minimalist perspective.[7] That said, the reader may have noticed that instances of external remerge may eventually lead to structures that *seemingly* involve tampering. I will come back to this shortly.

Now we know what the mergers as in (14) in (15) do *not* lead to, let us consider which structures they *do* create. The mergers in (16a–b) are familiar, as they involve *merge* or *remerge* at the root. The option in (16c) constitutes *external remerge*, which leads to a doubly-rooted graph:

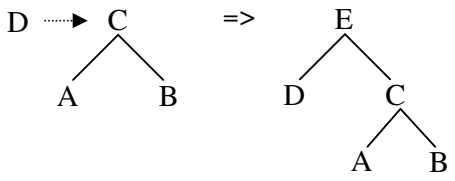
(16) a. Merge (D, [$_C$ A B]) → [$_E$ D [$_C$ A B]]
(regular *first-time merge*)

b. Merge (B, [$_C$ A B]) → [$_E$ B [$_C$ A B]]
(reg. *internal remerge*: mDom of B)

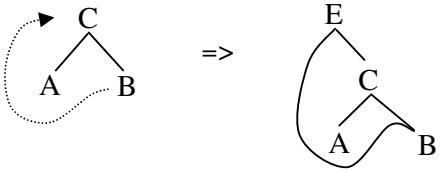
c. [$_C$ A B] and Merge (D, A) → [$_C$ A B] and [$_E$ D A]
(reg. *external remerge:* mDom of A)

[7] However, Chomsky (2000: 137), following Richards (1999), leaves open the possibility of 'tucking in' for 'third Merge', which would be a clear violation of the extension principle. The reason that this might be allowed is that it does not change the relationships of a *head* with respect to its complement and first specifier. Obviously, it does change basic relationships with and between projections of the head. It seems to me that in the absence of overwhelming evidence for tucking in, such a complication of Merge must be rejected. Merge, in its simplest definition, operates on syntactic objects, regardless their internal structure and projection status, creating lasting basic relationships between the input objects and the output object. See also section 4.

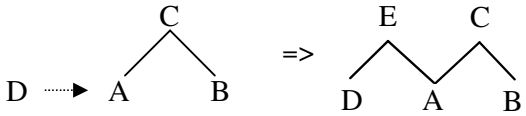
(16a)



(16b)



(16c)



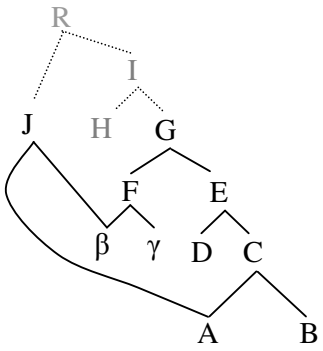
For completeness' sake, note that (16a) replaces (14a–b.i), (16b) replaces (15a.i), and (16c) replaces (14a.ii). The mergers in (15b.i) and (14b.ii) also involve regular internal and external remerge, respectively, and the actual resulting structures can be compared to (16b) and (16c), only then *remerge* concerns the other sister.

The problematic cases are the mergers in (15a.ii) and (15b.ii), which involve remerge with a non-root (namely, in (15a.ii), the term B is remerged with the term A; similarly, both V and B are embedded in (15b.ii)). The result cannot be structure-changing, as in (15), but instead an additional root node will be created, comparable to what happens in (16c). Consider a slightly more sophisticated and illustrative example. In (17), the problematic instance of Merge is accompanied by an exclamation mark. The first merger between brackets is a preparatory subderivation. The mergers in grey are a vain attempt to correctly finish the offensive structure.

(17) (Merge (β, γ) → F) (to be excluded)

Merge (A, B) → C
Merge (D, C) → E
Merge (F, E) → G
! Merge (A, β) → J

Merge (H, G) → I
Merge (J, I) → R



Here, A is remerged with the embedded β; this automatically leads to a temporary second root, J. The reason is that Merge by definition creates a new projection. (The same would apply if A were remerged with F itself, which is also embedded, namely in G.) Eventually, the two temporary roots can be combined into a final single root R, with possible additional material in between (such as H). One could call this 'quirky internal remerge' — internal, since no new

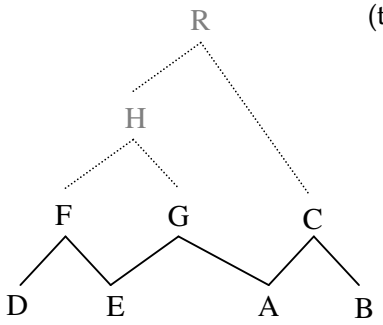
material is selected; quirky, because movement to an embedded position is normally considered ungrammatical. Furthermore, even if it were grammatical, the then intended string of abstract terminals is /H A β γ D B/, but I do not see how this could possibly be read off the structure. I conclude that the theoretical possibility of quirky internal remerge somehow needs to be excluded.

There is a counterpart of the above that we could call 'quirky external remerge', which involves remerge with an embedded position in another structure; see (18):

(18) (Merge (D, E) → F) (to be excluded)

Merge (A, B) → C
! Merge (E, A) → G

Merge (F, G) → H
Merge (H, C) → R



Here, A, which is a term of C, is externally remerged with E, which is embedded in the independently created F. As a result, a *third* temporary root, namely G, is created. Eventually, everything can be combined in one final root R, with possible additional material in between. The problem is that so far, I have not been able to come up with a realistic linguistic interpretation of (18). Moreover, like (17), it is clearly against the spirit of the extension condition, even though it does not involve 'tampering' in the strict sense.

Is there a plausible way to exclude both (17) and (18) at the same time? It seems to me that there is, but of course there is a theoretical cost to this, namely in the form of an explicit condition on the input for Merge. What (17) and (18) have in common is that at some crucial point of the derivation *both* input objects for Merge are terms and not roots (at that stage). A formal condition preventing this can be formulated as follows:

(19) *Root condition:*
If α and β are selected as input for Merge, then α or β (or both) must be a root.

There is a clear rationale for this condition. Consider (17) and (18) again. In both cases, the offensive instance of Merge creates an additional root where none was before. But this is not what Merge is for, from a functional perspective. Merge is essentially a combinatory device: it combines lexical items until a final single-rooted structure is created, which can then be pronounced. Bearing in mind that every lexical item itself is a root (of a trivial structure), Merge does the following:

A. If two lexical items are merged, the result is that the number of roots is reduced by one. Namely, after Merge (A, B) → C, where A and B are lexical items, the new root is C, and A and B have become terms of C.

B. For every other instance of first-time merge (which may involve complex items), the number of roots is reduced by one.
C. For regular internal remerge, the number of roots stays the same. Namely, if some term A (which is not a root) is remerged with the root R_i, R_i becomes a term of the new root R_{i+1}, and A is still embedded.
D. For regular external remerge, the number of roots stays the same. Namely, if some term A of root X is remerged with an independent root R_i, the result is that a new root R_{i+1} is created of which R_i is now a term. X remains a root, and A remains a term. So we start out with two roots (X and R_i), and end up with two roots (X and R_{i+1}).
E. For quirky internal remerge, the number of roots is enlarged by one. Namely, if some term A of root X is remerged with another term B of X, a new root R is created. Before merger, only X is a root; after merger, X and R are roots.
F. For quirky external remerge, the number of roots is also enlarged by one. Namely, if some term A of root X is remerged with some term B of another root Y, a new root R is created, and X and Y remain roots.

Thus, first-time merge is the best way to proceed towards the goal of creating a single-rooted structure.[8] Internal and external remerge are a necessary complication that causes some delay: The number of roots stays the same. But *quirky* internal/external remerge is completely counterproductive from this perspective, and must therefore be excluded. This insight is formalized in (20):

(20) *No proliferation of roots condition*

If the derivation proceeds from stage i to i+1 through Merge $(\alpha, \beta) \rightarrow \gamma$, then $|\{x \in \{\alpha, \beta, \gamma\}: x \text{ is a root at stage } i+1\}| \leq |\{x \in \{\alpha, \beta\}: x \text{ is a root at stage } i\}|$.

Informally stated: Upon Merge, the number of roots may not become larger.

The effect of (20) is completely equivalent to that of (19), so there are no two conditions, but just one that can be formalized in different ways, depending on the perspective. It is worth noting that there is a third way of looking at the root condition: one could conjecture that a derivation is always 'active at the top'.[9] Selecting a term is harmless as long as a root is involved as well. In the case of external remerge, the attention shifts from one structure to another. Quirky remerge does not involve any root in the input, and is therefore excluded.

The attentive reader will have noticed that quirky remerge was not included under the definition of internal/external remerge from the beginning — recall (4) and (5) —, and I will no longer consider it.

2.4. Remerge: A Discussion of Look-Ahead, Hydras, and Locality

Let us take RNR as a relevant example of a construction whose derivation

[8] This could also be taken as a rationale for Merge-over-Move effects, to the extent that these are real (see, e.g., Castillo, Drury & Grohmann, in press for an overview.).

[9] Compare also Collin's (2002) 'Locus Principle' (bearing on Chomsky's ideas about feature activity), which has largely the same effect, although it is not equivalent.

involves external remerge. A simple sentence such as (21), which is similar to (9) above, can be derived in the following way:

(21) Mary likes ___, and Jack hates *cars*.

1a Merge (likes, *cars*) → [likes *cars*]

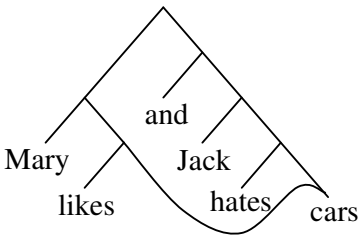
1b Merge (hates, *cars*) → [hates *cars*]

2a Merge (Mary, [likes *cars*]) → [Mary [likes *cars*]]

2b Merge (Jack, [hates *cars*]) → [Jack [hates *cars*]]

3 Merge (and, [Jack [hates *cars*]]) → [and [Jack [likes *cars*]]]

4 Merge ([Mary [likes *cars*]], [and [Jack [likes *cars*]]]) →
[[Mary [likes *cars*]] [and [Jack [hates *cars*]]]]



Here, step 1b involves external remerge of the direct object *cars*. During step 2a, 2b, and 3, the structure is doubly-rooted. Step 4 accomplishes a union into a single-rooted structure. Notice that there is no proliferation of roots at any step. In step 1b, we merge one root, [hates], with one term, [cars], and create one new root, [hates cars] — as merge always does. From the perspective of a syntactic workspace that initially contains all activated lexical items required for a particular derivation, we obtain the same result. In 1b, we start out with five roots, namely [Mary], [likes cars], [and], [Jack], and [hates], and we also end up with five roots: [Mary], [likes cars], [and], [Jack], and [hates cars].

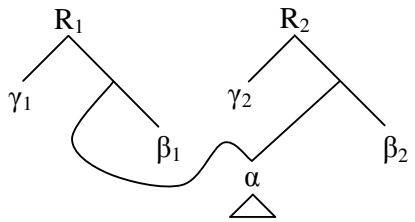
A number of other things are worth discussing. First, consider the order of mergers. The derivation in (21) seems to suggest that we start out merging *cars* in the first conjunct, and remerge it in the second. However, before the two clauses are conjoined, there *is* no first and second conjunct: The order between them (or their respective terms) is only established later in the derivation. Are we dealing with an instance of look-ahead here? By no means. It is of no importance whatsoever with which verb *cars* is merged first. The sequence of mergers in 1a/b and 2a/b can be switched around at will. Either permutation (1a–1b–2a–2b, 1b–1a–2a–2b, 1a–1b–2b–2a, or 1b–1a–2b–2a) leads to the same result. Therefore, it is impossible to tell which occurrence of *cars* — the one in the first conjunct, or the one in the second conjunct — is the original and which is the copy. As we said before, there are no copies, just relations. And there should be no need for pre-destination in syntax. It is for PF to decide where *cars* is to be pronounced, independently of how syntax arrived at the structure under consideration. Thus, if a particular structure has more than one possible derivational history, it should be pronounced the same in either case.

The absence of look-ahead implies that every instance of Merge must be motivated in some way or another. It does *not* imply that every structure that *can* be derived by Merge is interpretable at the PF/LF-interface. It is easy to think of licit derivations that are still uninterpretable in the end, that is, incomplete in some sense. For instance, a relevant feature could still be unvalued. Therefore, some possible derivations will survive at the interface, and some will not. This is not the consequence of look-ahead, even though it might seem so from the perspective of a surviving derivation. Within narrow syntax, Merge is an autonomous operation.

Turning back to the case of external remerge, we have noticed that it creates a multi-rooted structure (a 'hydra' or 'forest'). The particular step of Merge itself may very well be motivated: In example (21), the verb *hates* selects a direct object. However, the existence of more than one root is problematic for the linearization procedure at PF (see below for details). So it is convenient that the two clauses are conjoined at a later stage, which resolves the problem. We can derive a heuristic from this: *Every instance of external remerge must be compensated by a joining operation later in the derivation* (surely, this need not be coordination; it can also be parenthetical insertion or subordination). Such a heuristic may suggest look-ahead, but that is misleading. Obviously, the system itself has no meta-modular analytical intelligence. The preferred derivation will survive as long as both the instance of external remerge and the compensating joining instance of Merge are independently motivated within narrow syntax.

Consider the hydraic configuration in (22), where α is the sister of both β_1 and β_2 due to external remerge. The structure projects up to R_1 and R_2.

(22)



If this structure is sent to PF, how could it be linearized? The answer is that it could not at all. The reason is that the linearization procedure does not know where to start. And even if it randomly chooses one of the roots to be analyzed first, it is intuitively clear that no order between γ_1 and γ_2 can be established. With special additional assumptions, this may be resolved, but not in such a way that the order between the terminals remains invariant with respect to the choice of 'first root'. The two options here are the following: If R_1 is the root taking priority, the string of terminals will have to be /γ_1 γ_2 α β_2 β_1/; if R_2 is the root taking priority, the string of terminals will have to be /γ_2 γ_1 α β_1 β_2/. Clearly then, an asymmetry between the two (or more) temporary roots must be established: one is to be recognized as the matrix, the other as the secondary structure (a 'graft', using van Riemsdijk's terminology). The way to do this is to combine them in syntax. As a consequence, a graft cannot only involve sharing with a constituent of the matrix, the top of the graft must also be syntactically connected to the matrix (*pace* van Riemsdijk 1998, 2006). As I see it, the top connection is not only

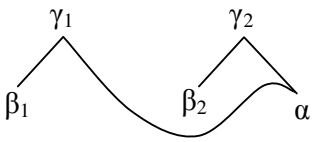
required because of PF demands, it also makes sense from a semantic and syntactic perspective. Namely, the way the graft is connected to the matrix determines the relationship between them. For instance, a graft can be a second conjunct, as in (21), or a parenthetical-like insertion, as in cleft-amalgams (10b), or perhaps even a subordinated phrase, as in parasitic gap constructions.

Next, let us turn to the issue of locality. In section 2.2., it was mentioned that RNR-constructions are insensitive to locality conditions, contrary to *wh*-movement constructions, for instance. This is illustrated in (23), where the dependency crosses (or seems to cross) the boundary of a complex noun phrase (with (23a) RNR and (23b) *wh*-movement our of a relative clause):

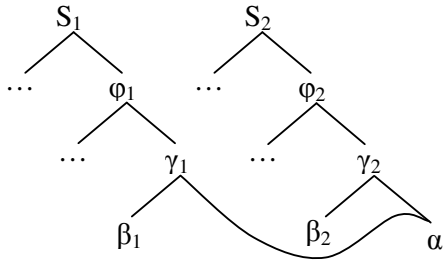
(23) a. Mary likes [men who SELL ___], but she hates [men who BUY *cars*].
b. **What* does Mary like [men who sell ___]?

External remerge, we said, creates a structural bypass. Let us see in a little more detail why this is so. Below, the derivation passes through the stages (24a–c).

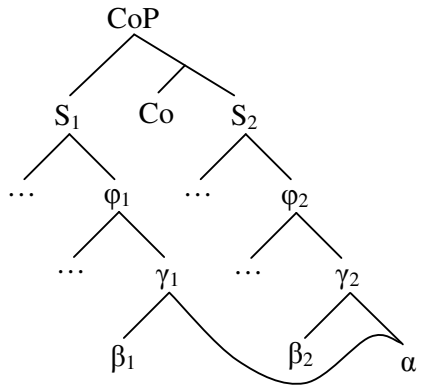
(24) a.



b.



c.



In (24), the constituent α is externally remerged. As no locality boundary is involved, yet, we will assume that this is unproblematic. In (24b), both spines of the structure are expanded by regular Merge up to S_1 and S_2. During this process, the locality boundaries φ_1 and φ_2 are created. In (24c), S_1 and S_2 are united in a coordination phrase. The end result gives the impression that α's relationship to

both β_1 and β_2 crosses the locality boundary φ. However, this is in fact not the case: α is locally related to β_1 and β_2 in step (24a). Whatever happens subsequently to this step cannot undo this local relationship. Put differently, the fact that β_1 and β_2 are not in the same local domain does not imply that some α cannot be locally related to both. This is the surprising consequence of external remerge.

If external remerge can create apparent locality violations in RNR, we may predict non-local behavior to show up in other types of sharing constructions as well. As far as I am aware, this has never been tested. I would like to claim that examples of this kind can indeed be construed. A relevant illustration in Dutch is a complex cleft-amalgam as in (25), where the parenthetical is to be interpreted as *de re*. The content kernel is italicized. Notice that a correct intonation is important: right before the dash, the pitch lingers relatively high in order to create a sense of expectation, the amalgam is pronounced relatively fast, and the content kernel is stressed.

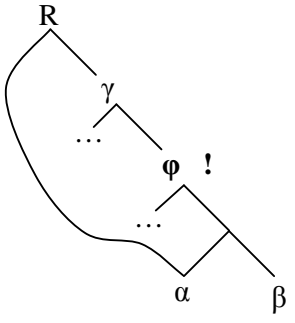
(25) *Dutch*

Joop kuste toen — Piet beweerde dat hij iemand kende die zei
Joop kissed then Piet claimed that he someone knew who said
dat het *Mieke* was.
that it Mieke was
'Then, Joop kissed — Piet claimed that he knew someone who said that it was Mieke.'

Within the amalgam, *Mieke* is embedded in a complex noun phrase. If it is true that *Mieke* is at the same time part of the matrix (namely, as a direct object), there seems to be a locality problem. But this is only apparently so, and the solution is similar to the one sketched above for RNR. For more examples (in English), see de Vries (2009b).

Does the possibility of bypassing locality boundaries not endanger our theory of locality for regular movement constructions? I do not think this is the case. Consider the following configuration:

(26) (to be excluded)



In (26), the phrase α is first-merged with β, and internally remerged with γ. If φ constitutes a locality boundary, the derivation is to be excluded. One way to do this is to make the selection of syntactic objects sensitive to structural distance. Thus, selecting a term as input for Merge is allowed as long as it is not too far embedded (where *too far* may be category-sensitive). From the perspective of

phase theory, if φ is a phase boundary, then it seals off its components for further computation. Thus, if the derivation in (26) has reached γ, α cannot be selected anymore since it is embedded in φ. Furthermore, a derivational bypass cannot be established, either: α cannot be externally remerged with γ before φ is reached for the simple reason that γ does not yet exist at that stage of the derivation. In section 3.5.3. I will come back to the issue of phases from the perspective of linearization.

2.5. Intermediate Conclusion

If terms of complex syntactic objects are allowed as input for Merge, *remerge* of heads and phrases is possible, as opposed to *(first-time) merge*. This is a way of dealing with the general phenomenon that an item can be involved in multiple (local) relationships, but shows up in only one position. Without ad hoc restrictions, it follows that there are two types of remerge: *Internal* and *external* remerge. The first corresponds to regular movement; the second is much more controversial, and has been characterized as sharing, grafting or sideward movement. Several construction types have been analyzed as involving what we now recognize as external remerge. Naturally, each of these will have to be subject to close scrutiny, and alternatives for some may be more viable in the end. However, what is of interest here is not so much the analysis of individual constructions, but the general mechanism of external remerge, in comparison to internal remerge. Both internal and external remerge can be represented in terms of multidominance. Though it has some graphical disadvantages, this prevents us from inadvertently attributing ad hoc properties to copies or traces. In this respect, it is worth commemorating that there is no inherent directionality in external remerge. If α is to be related to both β and γ, the order of mergers {Merge (α, β), Merge (α, γ)} is irrelevant, and look-ahead should not be necessary.

An automatic consequence of the structure-building characteristic of Merge is that it operates strictly cyclically: Merge creates a new root, and it cannot undo earlier relationships. But even then, some unwanted possibilities remain. These can be excluded by the *No proliferation of roots condition*, which seems a virtual conceptual necessity. As for locality conditions, they can be shown to be bypassed by means of external remerge in certain configurations, but this is never the case for internal remerge. Finally, we have seen that external remerge creates a temporary multi-rooted structure, which needs to be resolved before the structure gets linearized. As there are always asymmetric relationships between the different parts of the sentence, this will be taken care of for independent reasons as well.

3. The Linearization of Complex Syntactic Structures

Syntactic structures have to be linearized at PF. Structures exclusively composed of relations established by first-time merge are easy to process; the possibility of remerge, however, brings about some complications. This section advances a proposal for the linearization of structures involving internal and external

remerge. The discussion below will be in terms of multidominance graphs. We have to keep in mind, though, that graphs as such are only representations of the underlying relations between syntactic objects created by Merge. Section 4 will examine in some more detail the necessary linearization algorithm and the computational cost it involves.

3.1. The Problem of Remerge

Comparing *wh*-movement (27a) to RNR (27b), we notice a difference in the position where the displaced constituent (in italics) is pronounced:

(27) a. *Which violin* should this talented girl purchase ___?
b. The boy only admired ___, but the girl actually bought *this beautiful Stradivarius*.

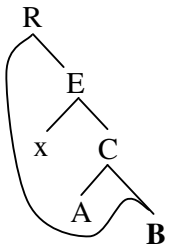
In (27a), the remerged phrase is realized in the first position in the string; in (27b) the remerged phrase is realized in the last position in the string. This must be due to the different effect caused by internal and external remerge, respectively. In fact, we are facing two complications:

(C1) Remerged items are only pronounced once.[10]

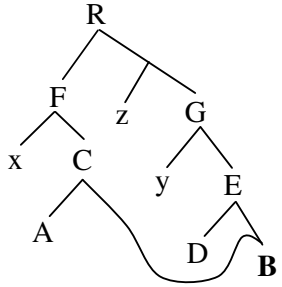
(C2) Internally remerged items are pronounced in a different position than externally remerged items.

The abstract configurations corresponding to possible derivations involving internal and external remerge are sketched in (28a) and (28b), represented in terms of multidominance:

(28) a.



b.



[10] I am aware of proposals involving spelled-out copies or traces for particular phenomena such as resumptive pronouns and *wh*-copying; for discussion, see Aoun & Li 2003, Grohmann 2003, Nunes 2004, Barbiers *et al.* 2008, and Schippers 2008, among others. This possibility, if correct, is of course exceptional. Moreover, I would like to stress that it does not present additional problems for a remerge approach to movement as compared to a copy/trace approach. Let me quote Starke (2001: 145) on this:

> Questions about multiple traces map onto questions about multiple mergers. Reduplication paradigms are another instance of this logic: To the extent that they are adequately analyzed in terms of spell-out of a trace (i.e. spell-out of multiple 'copies'), they are now reanalysed as spell-out of multiple merger operations […]. No novelty introduced there.

The constituent called B is displaced. It is remerged with E in (28a), and with D in (28b). In (28b), it could also be first-merged with D and remerged with A. Suppose we traverse the graphs in (28) in the usual way, starting at the root (I will come back to this in more detail), we encounter the remerged B twice. Putting the terminals in a string in the order we come across them, we obtain the following picture, where the intended Spell-Out is printed below the other strings. What are terminals in (28), by the way, need not be linguistic heads; only, their possible internal complexity is of no direct interest to us, here.

	(28a)	(28b)
Terminals encountered by graph traversal:	/**B** x A **B**/	/x A **B** z y D **B**/
Desired string of terminals:	/**B** x A/	/x A z y D **B**/

From this perspective, it is the first occurrence of B that needs to be overtly realized in (28a), but the second occurrence in (28b). The generalization can be stated as follows:

Internal remerge of X → overtly realize only the first occurrence of X while linearizing the structure.
External remerge of X → overtly realize only the last occurrence of X while linearizing the structure.

Below, I will discuss what happens if the two interact.

There is a body of literature on the linearization of syntactic structures involving movement, and there are also some publications on the linearization of sharing constructions (especially RNR). But as far as I know, the issue sketched here has not received any explicit attention in the literature, apart from Chen–Main (2006), Gracanin–Yuksek (to appear), and some brief remarks in Wilder (2008).

I have no intention of negatively reviewing other linearization proposals in detail — and no doubt each has its own merits —, but let me indicate in general terms why the course taken here is somewhat different. The most straightforward objection to all previous proposals I am aware of is the lack of general applicability: theories about movement (for instance, Fox & Pesetsky 2005) are unfit for sharing and *vice versa* (for instance, Wilder 1999); so the least we can say is that some adaptations are necessary.

Wilder (2008) and Gracanin–Yuksek (to appear) try to develop Kayne's (1994) Linear Correspondence Axiom (LCA) in order to manage sharing [external remerge] (see also Johnson 2007 for discussion), and briefly investigate if some extended version of the LCA is also fit for movement in terms of multidominance [internal remerge]. As they note themselves, it runs into trouble with moved constituents that are complex, that is, simply phrasal (Wilder), and with shared material that is not the most deeply embedded (Gracanin–Yuksek). Apart from that, Wilder explicitly derives the right periphery condition on RNR from the linearization procedure. To the extent that this is successful (see however Kluck

& de Vries, to appear, for critique, which also applies to Bachrach & Katzir 2009), it may be considered unfortunate, since it prevents the generalization of the idea of structure sharing (external remerge) to other constructions in which the presumed shared part is not necessarily right-peripheral.[11] Possible examples involve what van Riemsdijk (1998) called 'saddle grafts', where the shared material is not peripheral inside the graft. A cleft-amalgam in Dutch is (29):

(29) *Dutch*

Joop	heeft	[ik	vermoed	dat	het	*een*	*Bugatti*	is] gekocht.
Joop	*has*	*I*	*presume*	*that*	*it*	*a*	*Bugatti*	*is bought*

'Joop bought — I presume it's a Bugatti.'

A further construction that is interesting from this perspective is ATB-movement, where external remerge seems to feed internal remerge (see Nunes 2004 and Citko 2005, among others). In the Dutch example in (30), the two gaps represent indirect object positions; these are certainly not clause-final:

(30) *Dutch*

Wie	heeft	hij	___	een	boek	gegeven	en	zij	___ een	cd ontnomen?
who	*has*	*he*		*a*	*book*	*given*	*and she*		*a*	*CD taken.away*

'Who did he give a book and she take away a CD from?'

Citko explicitly states that movement following sharing is necessary for successful linearization at PF. Her theory, therefore, is quite limited, since it excludes an analysis of RNR and amalgams in terms of sharing.

Chen–Main's (2006) analysis is the most extensive one; it combines an inherent asymmetry between sisters (precedence) with LCA-like demands on the linearization. The proposal has the following basic characteristic: Multidominance of left branches leads to the pronunciation of the highest, leftmost occurrence, whereas multidominance of right branches leads to the pronunciation of the lowest, rightmost occurrence. The first corresponds to regular movement, the second to sharing in RNR-constructions. A serious problem discussed by Chen–Main herself is *wh*-movement of direct objects, which is excluded by the system. She cleverly turns this into a partial advantage by using it to explain Holmberg's generalization: Namely, the configurational problem is resolved if the verb is moved as well. But of course Holmberg's generalization is far from universal; moreover, I think the problem concerns movement of right branches in general, not only of direct objects. Chen–Main discusses various complicated interactions between multidominance links in syntactic graphs. It turns out that some structures can be linearized, some cannot, and some are linearized

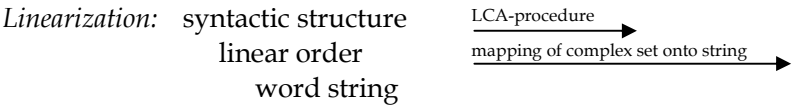
[11] The particular right-periphery condition for RNR must then be explained in another way. In fact, this conclusion is corroborated by Kluck (2007, 2009), who shows that purely syntactic approaches to derive the periphery effect fail; instead she proposes to combine Hartmann's (2000) theory on the semantics and prosody of the RNR-construction with a multidominance approach. Put differently, the periphery condition is an interface effect clearly related to contrastive focus; a successful explanation must somehow take that into account.

in a way we would not expect. This raises a more fundamental question: Do we want phonology to restrict syntax in such an intricate and hard-to-predict way? Without lessening any of the value of Chen–Main's discussion, my answer right now would be negative.

This answer confirms my general skepticism towards LCA-based analyses. Let me briefly formulate some general considerations indicating why I am not convinced that this kind of approach is attractive, apart from the issues mentioned above. I should mention beforehand that this does not imply that I am against a universal Spec–Head–Comp order, and hence right-branching graphs. The reason is that this universal is based on conceptual reasoning as well as empirical generalizations. In the absence of a phonological principle such as the LCA, it could simply be hard-coded in syntax.

First, the LCA presupposes that syntax does not directly encode an asymmetry between sisters that can be interpreted as precedence by phonology. Therefore, the necessary asymmetry at PF must be calculable from hierarchical information. However, the mentioned presupposition has been questioned in the literature. In section 3.2.1., I will come back to the idea that Merge produces ordered pairs.

Second, the LCA is often used to linearize (spell out) structures, but strictly speaking, the LCA states a necessary *property* of syntactic structure; it is not a *procedure* to arrive at the demanded (transitive, antisymmetric, and total) linear order from the syntactic structure. Kayne (1994) explicitly formulates the LCA as an *axiom*, not an *algorithm*. Nevertheless, suppose that we formulate such a procedure on the basis of the LCA (see section 4 for an impression how to construct a linearization algorithm). Even then we are not there, yet: What PF wants is not a mathematical linear order, but a string of words. Of course, we can formulate another mapping procedure that translates the set of ordered pairs of terminals that constitutes a linear order into a string, but it should be clear that this is an additional step:



From a practical perspective, if we can go directly from syntactic structure to a word string, this seems preferable over the state of affairs sketched above.

Third, the LCA involves a very intricate definition of c-command. In the extended approaches it is even further enriched by the notion of full dominance/ unique paths (see Wilder 2008, for instance). But c-command is a very general syntactic tool, used for many more things than linearization; essentially, it identifies possible dependencies. It is doubtful that such a fundamental notion could be so complex. Rather, it seems likely that c-command is a direct function of Merge, as initially proposed by Epstein (1999) — put briefly, if A and B are merged, then A c-commands B and every term of B.

Fourth, c-command is restricted to full categories (heads or maximal projections); segments (or X-bar nodes, in Chomsky 1995) do not count. As a consequence, a specifier asymmetrically c-commands the components of its sister,

but not *vice versa*. It seems to me that if the asymmetrical behavior between specifiers and their sisters needs to be (indirectly) stipulated in this way, the goal of deriving linear order from hierarchy is not convincingly attained.

Fifth, the LCA is hard to combine with Bare Phrase Structure; see Chomsky (1995), Uriagereka (1999), and others. In particular, there is a lack of asymmetric c-command between the very first two (lexical) items that are Merged in a derivation.

Sixth, given that movement is now viewed as internal remerge, configurational complications arise if more than one phrase is remerged. If I am not mistaken, LCA-based proposals inevitably run into trouble with remnant movement and roll-up movement (see section 3.5.1. for abstract illustrations of such constructions).

In the next sections, I will formulate an alternative approach that does not suffer from the limitations discussed above. It is fair to say, though, that it is not free of stipulations that are in want of a deeper explanation. My main objective here is to make the required linearization procedure fully explicit.

3.2. Linearization as a Process: Theoretical Preliminaries

Linearization is the process of turning a hierarchical structure into a string of terminals. In line with Kural (2005), Kremers (2009), and others, I assume that the most straightforward way to do this is by means of graph traversal. The details of this process will be discussed from section 3.3. onwards. But first, a number of theoretical preliminaries need to be addressed.

3.2.1. Asymmetrical Syntax

Graph traversal implies that the linearization procedure can make use of the precedence relation between sisters (known as *direct precedence*, *immediate precedence*, *strict precedence*, or *sister precedence*). This is in line with Frampton (2004), Chen–Main (2006), and many others, now, and especially in the 1980s, when the idea that directionality between heads and complements was considered a language parameter by most people. However, it goes against Kayne (1994) and Chomsky (1995), who state that syntax is about hierarchy and not about order. From the perspective of a derivational grammar, this means that Merge produces a complex whose components are unordered (an unordered set, according to Chomsky).

However, even when we grant the idea that syntax should not be preoccupied by linear order between sisters, the conclusion that Merge produces an unordered pair does not logically follow. After all, it is very well possible that Merge produces an ordered pair that encodes a syntactic or semantic asymmetry. If this is so, it may be the case that the relevant syntactic asymmetry can directly be mapped onto direct precedence at the PF interface. But that does not mean that linear order is part of syntax. It is simply a misconception to equate asymmetry between sisters in syntax with linear/temporal precedence, even if it will eventually lead to this in the phonological component.

So now we face two questions: First, is there indeed a consistent syntactic

or semantic asymmetry between sisters? In other words, does Merge automatically produce ordered pairs? Second, can this syntactic asymmetry consistently be mapped onto direct precedence at PF? The first question has been answered positively by several authors, albeit on somewhat different (but not necessarily contradictory) grounds: Jaspers (1998), Koster (1999, 2003, 2007), Di Sciullo (2000), Langendoen (2003), Zwart (2004, 2006a), Di Sciullo & Isac (2008). If the second question is to be answered positively, a much-preferred condition, I presume, is a universal base order. The most likely is a universal Spec–Head–Comp order, as advocated by Kayne (1994), Zwart (1994), and many others since (*pace* alternative configurations in Fukui & Takano 1998 and Haider 2000). In accordance with several of the cited works, let me sketch an account in terms of *dependency*.

The operation Merge creates sisters. From a compositional-semantic point of view, the objective of Merging, say, X and Y is to relate X to Y, thereby giving rise to a combined meaning. Crucially, sisters are never in a symmetrical relationship. For one thing, it is generally assumed that complements are c- and s-selected by heads. If it is indeed the case that complements universally follow heads, then the PF-mapping of this syntactic asymmetry on a phonological direct precedence relationship is unproblematic.

How about the combination of phrasal constituents? According to Koster's *configurational matrix*, there is always a left–right asymmetry (presupposing universal SPEC-left). Semantically, the righthand sister is 'relatively about' the lefthand sister, which is normally more salient: For instance, the predicate is interpreted with respect to the subject; a comment is about the topic, and so on. Syntactically, anaphoric dependencies (in the broadest possible sense of the word) are from the lefthand sister to (a term of) the righthand sister, *modulo* reconstruction effects due to A-bar movement, as is well-known. Thus, in a configuration $[_{\gamma}\ \alpha\ \beta]$ it is always β (and indirectly, its terms) that is dependent on α. Zwart's (2006a) hypothesis is that dependency is a function of Merge. Thus, Merge (α, β) produces an ordered pair <α, β> such that β is the dependent. If Zwart and Koster are right, then the asymmetrical merger of phrases can be mapped onto direct precedence at the PF interface, as well. The generalization is that if α and β are merged, whether they are heads or phrases, then the direction of dependency can directly be mapped onto direct precedence, such that the dependent always follows the non-dependent. Let us call this the Uniformity of Mapping hypothesis:

(31) *Uniformity of Mapping Hypothesis*

At the PF interface, generalized syntactic dependency is directly mapped onto phonological precedence, such that in a basic syntactic triad $<_{\gamma}\ \alpha, \beta>$, α will directly precede β.

In the next sections, I will use (31) as a background assumption. In principle, the necessary mapping could as well be performed by a more intricate rule system that makes reference to language-specific directionality parameters, but let us stand by the simplest solution.

Before we return to the main subject, a note on the widespread idea of Spec–Head agreement may be in order. Crucially, a specifier is a sister of a

projection of the head: In [$_{XP}$ Spec [$_{X'}$ X Comp]], Spec is the sister of X′. Since a projection of X contains all the features of X per definition (this is the idea of percolation), the Spec–Head relation can be reduced to a sisterhood relationship. One could say that the head is the nearest (most local) term of the sister of the specifier, which is probably the reason why the Spec–Head relation is important. Therefore, it comes as no surprise that it is often morphologically encoded. Nevertheless, there are indications that Spec–Head is only a special case of the more fundamental sisterhood relationship. An interesting example in this respect is subject-predicate agreement in Swahili. Consider the example in (32), taken from Carstens (2003: 395):

(32) *Swahili*

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 morpheme is spelled out on several elements within the predicate, including an adverb. Zwart (2006a) suggests that we can analyze this example as follows: Not just the finite verb, but the predicate as a whole, being the sister of the subject, is marked as the dependent of the subject. This dependency can then be spelled out on several terms of the predicate; which one(s), that concerns a language-particular morphological choice.

A line of research with very similar characteristics is Matushansky's (2008) approach to Case marking. In her view, Case marking involves a sisterhood dependency, which may eventually be spelled out on a term of the dependent sister (normally a DP). In this way, different Case features may accumulate on a single DP, which leads to language-particular morphological choices.

Finally, let me briefly comment on Chomsky's conception of set-Merge. Chomsky (1995, and subsequent work) 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. At first sight, the issue of the label is interesting in the light of the by now famous Wiener–Kuratowski convention (Wiener 1914, Kuratowski 1921), which states that an ordered pair <x, y> is equivalent to {{x}, {x, y}}. This complex set is often thought to be equivalent to {x, {x, y}}, for instance in Cormen *et al.* (1990: 80); see also Quine (1945) and Schneider (1977) for discussion. Such set-theoretic reductionism has been criticized by several philosophers (see Armstrong 1986, Forrest 1986, Goodman 1986, Sider 1996), mainly because it is arbitrary. Nevertheless, the convention is widely used. When applied to Chomsky's notation, it would follow that {a, {a, b}} is an ordered pair <a, b>. Thus, one might suppose that Merge (a, b) produces the ordered set <a, b> in case *a* projects, and <b, a> in case *b* projects. We will see that this is of no advantage for the linearization at PF.

Suppose we map the asymmetry of projection onto direct precedence at PF. That would produce an unfortunate result: Either we obtain a universal order /Head Comp Spec/ (if projecting elements precede non-projecting elements) or we obtain /Spec Comp Head/ (if the reverse is the case). The first is assumed by

no one; the second is actually proposed by Fukui & Takano (1998), but it is in clear contrast with assumptions by Chomsky himself, and of course Kayne (1994); see also Yasui (2004) for discussion on related issues, and Zwart (2005), who provides additional evidence for a universal /Head Comp/ structure based on a cross-linguistic typology of noun phrase conjunction. I conclude that the asymmetry of projection is not the asymmetry we are looking for. In this respect, recall that it is not certain that {x, {x, y}} can be equated with <x, y>. Also, I am not convinced that Chomsky is right that Merge (x, y) produces {x, {x, y}} to begin with, apart from the issue of asymmetrical dependency discussed above, and the question if projection labels are necessary at all (Collins 2002). The idea that the projection label equals the head is strange from the perspective of compositionality. Logically, the whole cannot equal one of the parts. One might, not unreasonably, object that the label x is not the same as the head x: These are different categories of things. But then of course set reduction of $\{x_{label}\ \{x_{head}, y\}\}$ to <x, y> is impossible.

In short, I contend that Merge produces asymmetrical pairs that encode syntactic dependency. This asymmetry can be mapped straightforwardly onto direct precedence at the PF interface. If this is so, one might wonder why we still need a linearization procedure involving graph traversal. The reasons are clear-cut. Even though a simple structure <a, <b, c>> transparently yields the string /a b c/, it should be noted that this does not directly follow from the transitivity of precedence, as it is mediated by the inclusion relationship. The next subsection shows that the usually implicit 'Inheritance of Precedence' assumption is untenable as soon as we take *remerge* into account (whether for regular movement or for sharing). Syntactic structures are not simple trees, they are graphs. Furthermore, recall that establishing a (total) order, for example, {{b<c}, {a<b}, {a<c}}, where < means *precedes* — is only halfway the goal of producing an actual word string (see also the second objection against the LCA near the end of section 3.1.).

3.2.2. Dominance/Inclusion and Precedence

Merge (α, β) → γ produces three relationships: γ directly includes α, γ directly includes β, and β is directly dependent on α. At PF, the last relationship can be reinterpreted as α directly precedes β.

Both inclusion and precedence are transitive relations: If αRβ and βRγ then αRγ. Both are irreflexive: αRα is excluded. I assume the notion *dominance* to be synonymous with inclusion. Reflexive dominance/inclusion plays no role in the discussion here, and I will abbreviate *proper dominance* to dominance.

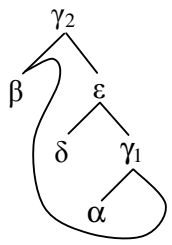
Direct precedence and direct dominance resulting from the same instance of Merge are mutually exclusive: if α directly precedes β, then it cannot be the case that α directly dominates β or *vice versa*. It is often assumed that precedence and dominance *in general* are mutually exclusive. Whether this is really the case, however, depends on further assumptions. Such an assumption is the inheritance of precedence, which can be formulated as in (33):

(33) *Inheritance of Precedence* (to be rejected)

If x directly precedes y, then x and all nodes dominated by x precede y and all nodes dominated by y.

In a regular tree, this makes sense. However, in multidominance graphs, it gives rise to inconsistencies. Indeed, versions of (33) are known as the Non-tangling condition, which is used to prevent crossing branches and so on (see also Gärtner 2002, Carnie 2008, and Fortuny 2008 for further discussion and references). Consider (34), where β has been internally remerged:

(34)



Here, β directly precedes ε; by (33), β would also precede the descendants of ε, namely δ, γ_1, α and β. But then β precedes itself, which is odd. Even worse, δ precedes γ_1 and, by (33), α and β. Thus, β precedes δ (the descendant of ε), and δ precedes β (the descendant of γ_1), which is a contradiction. Moreover, notice that ε and γ_1 dominate β *and* are preceded by β at the same time. Given that (34) is a regular movement structure (involving internal remerge), it seems to me that we must simply reject the stipulation in (33). I conclude that the precedence relation is independent of the dominance relation. Therefore, there is also no basis for the mutual exclusion of dominance and precedence in general.

As we have seen in section 2, the absence of stipulative conditions on the input of Merge leads to the possibility of remerge, which in turn leads to multi-dominance representations. Traditional restrictions on possible tree representations, such as the Single mother condition and the Non-tangling condition, are no longer wanted. It should be clear that it is not the case that anything goes. For instance, Merge makes sure that the resulting graphs are fully linked with respect to (transitive) dominance in the sense that there is a path from every node to every other node (that is, if we allow for a 'change of direction'). Furthermore, we saw in section 2.3. that the PF interface cannot possibly interpret multi-rooted structures (forests); thus, the traditional Single Root condition will be maintained on principled grounds (but notice that this only concerns the end result of the derivation). Also, I argued that the proliferation of roots during the derivation is unwelcome (section 2.3.).

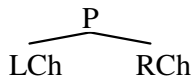
In the previous section, an LCA-based approach to linearization was abandoned on principled and practical grounds. Instead, let us try to develop an alternative in terms of graph traversal, which is inherently a procedure. In the next sections, let our guiding principle be the following: *Every single-rooted structure that can be produced by Merge can be linearized*. In other words, grammatical syntactic structures do not crash when they are linearized at PF.

3.3. Tree Traversal

Before we turn to multidominance graphs, let us have a look at tree traversal, which is less complicated. A standard order-sensitive top–down depth-first

recursive tree traversal procedure yields a string of node contents. First, consider the basic triad in (35), were P is the root, LCh the leftmost child, and RCh the rightmost child.

(35)



As for the terminology, note that what linguists often call *mother* is also known as *parent* or *(immediate) ancestor*; *daughter* is also known as *child* or *(immediate) descendant*.

In principle, there are three ways of traversing (35): (i) the so-called *preorder* traversal, which lists the parent first, and then the children from left to right; (ii) the *inorder* traversal, which lists the leftmost child first, then the parent, and then the rightmost child; and (iii) the *postorder* traversal, which lists the children before the parent.

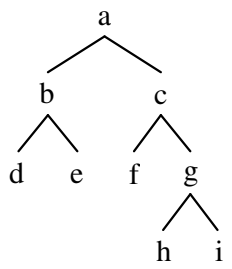
Tree traversal is a recursive algorithm, consisting of three basic steps (in a binary tree): Select the leftmost child, select the rightmost child, and perform some action, such as listing the present node content. If a complex child is encountered, interrupt the activity in the present layer and start scanning the child first, returning to this higher layer later (this is called recursive depth-first scanning). The core of this procedure is stated in (36), the three possible positions of undertaking the action are indicated.

(36) *Scan Triad*

↓ (← list present node: Preorder)
↓ select left child; if complex => start *scan triad* of child
↓ (← list present node: Inorder)
↓ select right child; if complex => start *scan triad* of child
↓ (← list present node: Postorder)

The results of scanning the more complicated abstract tree in (37) are given in (38).

(37)



(38)

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')

A linguistic linearization requires a list of end nodes (terminals) only. In (39), the strings from (38) are repeated, with the terminals printed in boldface.

(39) 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/

Interestingly, 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: the difference between preorder, inorder, and postorder is irrelevant in practice.[12] What we do is recursively call upon a procedure like *scan triad*; the action of spelling out is restricted to terminals. It seems to me that this is a welcome conclusion.

Just to be concrete, let me describe precisely what happens if we linearize the tree in (37). We start at the root, which is *a*. This node has children; we turn to the preceding one, *b*, first; *b* is complex as well; we turn to child *d*, which does not include members. The content of node *d* is therefore added to some string that is initially empty. 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 procedure. The obtained string is /d e f h i/, as required.

In conclusion, traversing a tree in order to produce a string of terminals is straightforward. No puzzling stipulations are necessary.

3.4. Traversing Multidominance Graphs: The Issue of Internal and External Remerge

Let us now turn to the effect of remerge on the linearization. It will become clear that we have to combine traversal with structural conditions. In section 3.1., we saw that a remerged node is encountered twice when traversing the graph at PF, whereas it is only pronounced once. Dealing with regular movement, Frampton (2004) proposed the following structural condition:

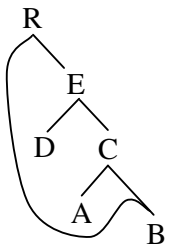
(40) The linearization of x (x = α or β) in $_{\alpha}/^{\gamma}\backslash_{\beta}$ is omitted if x has a parent outside γ. (to be revised)

Here, *outside* means 'not dominated by'. The effect is that α is pronounced in its highest position. From this perspective, consider the graph in (41):

---

[12] 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. See also Kural (2005) for a proposal in which the difference between preorder, inorder, and postorder traversal for non-terminals is exploited to account for word order variation.

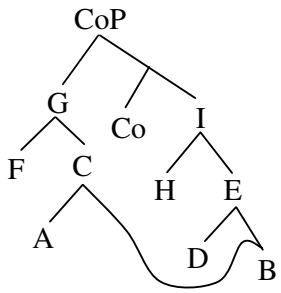
(41)



The remerged node is B. Its two parents are C and R. Traversing the graph, we arrive at B directly from R. As B has no other parent outside of R, the linearization is *not* omitted; hence, let us assume that the first occurrence of B is linearized (spelled out). We go on, and spell out the nodes D and A. Then we arrive at B for the second time, now from C. In this case, there *is* another parent outside C, namely R; therefore, the linearization is omitted the second time, as required. Setting aside the possible internal complexity of A, B, and D for a moment, the produced string is /B D 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 an RNR-configuration is given in (42), where the intended string of terminal syntactic objects is /F A Co H D B/:

(42)



Let us see what happens if we apply the condition in (40). The relevant remerged 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 outside C. This is the case: There is another parent, E, which does not dominate C. 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.

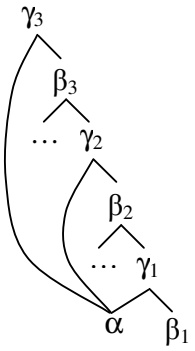
How can we improve on Frampton's condition? We have to take into account several things; so let us proceed step by step. As a first preliminary, let us change perspective from omitting the linearization of some node to spelling it out. After all, in a linearization procedure (and more generally), we would rather want to know under which conditions a certain action is to be performed than when we have to do nothing. The absence of events is a universal default; actions need to be specified. In language, successive cyclic movement and multiple RNR show that silence rather than pronunciation is the default:

(43) a. *What* did John say ___ that Mary thought ___ that Bill ___ bought?
b. John hates ___, Bill likes ___, Mary admires ___, and Jack detests *the president*.

In principle, only one occurrence of a linguistic object is overtly realized, whereas the number of silent occurrences is unbounded.

Furthermore, during the traversal, we have to keep track of where we came from. Otherwise, we do not know what a potential *other* parent is. Consider (44), where α has been internally remerged twice (its three parents are γ_1, γ_2, and γ_3; its sisters are β_1, β_2, and β_3):

(44)



At some point during the traversal the node under consideration is α. How is it determined if α is to be spelled out at this point? That depends on the path from which α is most recently arrived at. Node α must be spelled out in the triad $[_{\gamma 3}\ \alpha\ \beta_3]$, but not in $[_{\gamma 2}\ \alpha\ \beta_2]$ and $[_{\gamma 1}\ \alpha\ \beta_1]$ because there is a parent, γ_3, that is outside (in fact, dominates) γ_2 and γ_1. But how do we know in which triad we are at the moment? Each parent γ_1, γ_2, and γ_3 is equally local to α. Therefore, we need to keep track of the traversal history in some way. To this end, let us define the notion of *current parent*:

(45) *Current Parent*

The current parent of α is the most recently traversed parent during the linearization procedure.

We can now reformulate (40) as follows:

(46) *Spell-Out of (Internally) Remerged Nodes* (preliminary version, to be revised)

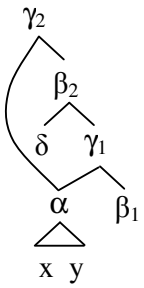
Linearize an α with more than one parent if the current parent dominates every other parent.

Here, I take *dominance* to be a transitive, non-reflexive relation; an *other parent* is a parent that is not the current parent. Note that the condition in (40) has much in common with the 'Connected ancestor condition' discussed in Barker & Pullum (1990: 22).

We have to be aware that *Spell out/Linearize* α is not always equivalent to

adding α to the string to be pronounced, although it can be. This has to do with the difference between heads and phrases. A phonological string of words or morphemes is a string of heads. So if α is complex, it must be analyzed by further traversal. Of course movement often concerns phrases; therefore, consider (47):

(47)



Here, the phrase α containing two heads x and y has been internally remerged; the intended string of terminals is /x y δ β_1/. Without a condition such as (46), x and y are encountered twice during the traversal. This would be problematic, since x and y themselves are relatively *in situ* heads (with only one parent), so their Spell-Out should be uncompromised, which leads to a double realization. According to (46), the linearization of α in (47) is executed only if γ_2 is the current parent. If γ_1 is the current parent (later during the traversal), α will not be spelled out, which implies that its components will not be traversed; hence, x and y are not encountered a second time.

Before we turn to external remerge, let me summarize the assumptions so far:

(A1) The linearization of a syntactic object involves recursive graph traversal.

(A2) Traversal history needs to be monitored.

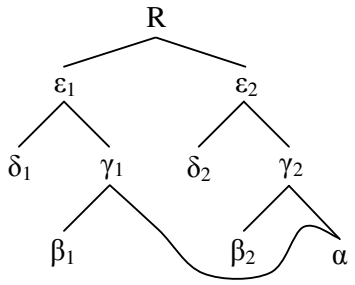
(A3) For each encountered node, further analysis of its components is conditional: If the number of parents is zero (for the root) or one, further analysis is called upon in any case; if the number of parents is more than one (the consequence of remerge), the configuration between these parents, relative to the current parent, is decisive.

(A4) Further analysis means further traversal if the relevant object is complex. If it is not, the node content is added to the string of words/morphemes.

(A5) Traversal provides a continuously shifting perspective, where the current node and indirectly its current parent are the center of attention. But notice that the interaction with structural conditions implies that the rest of the structure can be inspected at any time.

We are now in a position to analyze (48), which involves external remerge of α. Its parents are γ_1 and γ_2; the intended string of terminals is /δ_1 β_1 δ_2 β_2 α/:

(48)



Each of the two parents is 'outside' the other; none dominates the other. The configuration seems symmetrical, but not if the course of the traversal is taken into account. Since we already established before that the traversal history needs to be monitored, I take this to be a possibility. The crucial decision depends on the direct precedence relationship between two of the ancestors of α, namely ε_1 and ε_2, which have been the input for the 'root-uniting' instance of Merge. Parent γ_1 is part of the left branch of the graph, which is traversed before γ_2 in the right branch of the graph. When α is encountered the first time during the traversal, via γ_1, the other parent γ_2 has not yet been traversed; however, when α is encountered the second time, via γ_2, the other parent γ_1 has already been traversed. Therefore, the necessary Spell-Out condition can be formulated as follows:

(49) *Spell-Out of (Externally) Remerged Nodes* (preliminary version)
Linearize an α with more than one parent if
(i) every parent has been traversed, *and*
(ii) the current parent is not dominated by any other parent.

As required, α will be spelled out if γ_2 is the current parent. The second proviso in (49) is necessary to make sure that Spell-Out of the last occurrence is not applied to configurations resulting from *internal* remerge, where one parent dominates the others. Let us now combine the two conditions in (46) and (49):

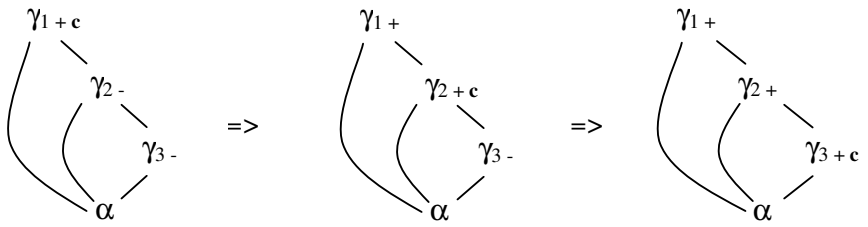
(50) *Spell-Out of Remerged Nodes* (to be revised)
An α with more than one parent is linearized if and only if
(i) the current parent dominates every other parent; *or*
(ii) every parent has been traversed, *and*
the current parent is not dominated by any other parent.

No contradictory linearization demands can be imposed on a multidominated node. First, consider some basic possible configurations. In each case, α is the shared node, γ_i is a parent, a subscript *c* indicates the current parent at a particular stage of the traversal, and the plusses and minuses indicate which parents have been traversed at this stage. For ease of exposition, all other sentence material is omitted.

The repeated structure in (51) represents movement via an intermediate landing site, which results from applying internal remerge to the same syntactic

object twice:

(51)

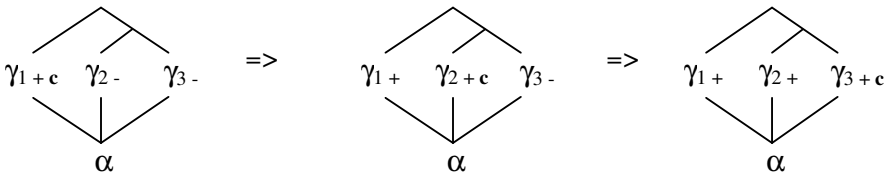


spell out α omit α omit α

The three structures correspond to different stages where α is reached during the traversal. The decision whether it spelled out at this point is printed below the structure. In the first structure, we arrive at α from the highest parent γ_1. According to (50i), α is spelled out; (50ii) does not apply because the other parents have not been traversed yet. In the second and third structure, α is arrived at from γ_2 and γ_3, respectively, and the linearization of α is to be omitted. Indeed, (50i) no longer applies, since γ_2 and γ_3 are not the highest parent; and (50ii) does not apply because they are dominated by γ_1.

The case of external remerge is sketched in (52). Again, α is remerged twice (a concrete example could be (43b) above). Recall that all parents $\gamma_{1/2/3}$ are locally related to α, which is the center of attention three times during the traversal.

(52)

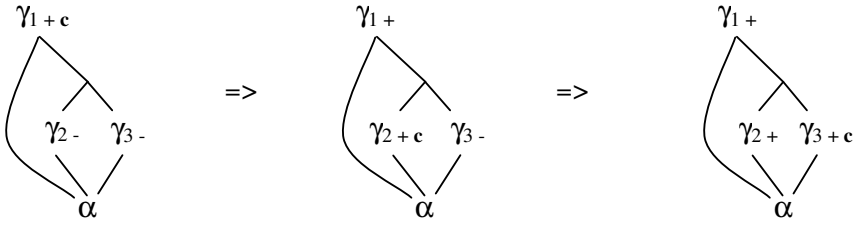


omit α omit α spell out α

In the first two situations, γ_1 and γ_2 are the respective current parents. They do not dominate all other parents, so (50i) does not apply. Furthermore, not every parent has been scanned, yet, so (50ii) does not apply either, and the linearization of α is omitted. In the third situation, where α is encountered the third time, now via γ_3, every parent has finally been traversed, and α is spelled out.

We also have to check what happens when internal and external remerge are combined. There are two basic possibilities. The first is pictured in (53), where a sharing configuration is embedded in a movement configuration; this results from internal remerge, after root-union, of a constituent that has already been externally remerged. Concretely, this may correspond to ATB-movement (see section 2.2.).

(53)

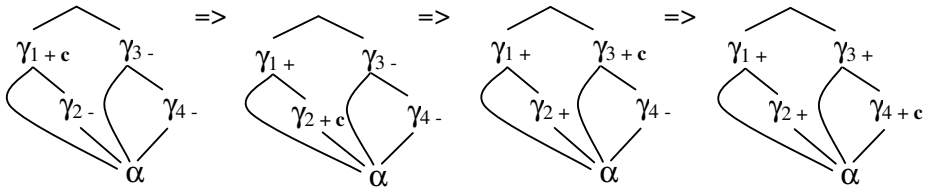


spell out α omit α omit α

Since there is one parent, γ_1, that includes all other parents of α, the structure will be handled on a par with standard movement configurations as in (51), and α is spelled out in the first position accessed, as required.

The second configuration is pictured in (54). Here, internal remerge is followed by external remerge, or at least root union has to apply after internal remerge. Intuitively, it seems clear that the third occurrence of α needs to be spelled out. This is the position where α is moved to inside the relevant dependent substructure. For symmetry reasons I have added internal remerge in the matrix (or first conjunct) as well (a concrete example may be a 'saddle graft' such as (29), where the object is moved to the middle field.)

(54)



omit α omit α spell out α ?! omit α

However, according to (50), α will never be spelled out: (50i) never applies since there is no parent that dominates every other parent (note that γ_1 dominates γ_2 but not the other parents; similarly, γ_3 dominates γ_4 but not the other parents), and (50ii) never applies because only when γ_4 is the current parent, all parents have been traversed, but γ_4 is a parent that is dominated by another parent (γ_3). How can this omission be repaired? The answer is that (50i) must be relativized according to the traversal status of the dominated nodes, as is shown in (55). The second proviso in (55i) is necessary to prevent the Spell-Out of α in intermediate landing sites.

(55) *Spell-Out of Remerged Nodes* (correct, pre-final version)

An α with more than one parent is linearized if and only if

(i) the current parent dominates every other parent <u>that has not been traversed</u>, *and*
the current parent is not dominated by any other parent; *or*

(ii) every parent has been traversed, *and*
the current parent is not dominated by any other parent.

In (54), α will now be spelled out if the current parent is γ_3, since this parent dominates all parents not yet traversed, namely γ_4 alone. Crucially, the first occurrence of α, via γ_1, does not lead to Spell-Out, since at this point of the traversal, γ_3 and γ_4 are not yet traversed, and they are not dominated by γ_1.

The second proviso in (55i) equals the second proviso in (55ii); therefore, a more compact formulation is possible:

(56) *Spell-Out of Remerged Nodes* (final version)

An α with more than one parent is linearized if and only if

(i) the current parent is not dominated by any other parent, *and*

(ii) – every parent has been traversed, *or*
– the current parent dominates every other parent that has not been traversed

This concludes the basis of the proposal. The next section discusses some complex cases and potential problems.

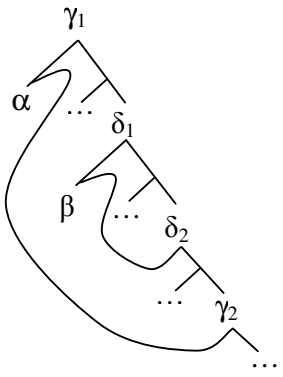
3.5. Complex Structures and Potential Problems

In a number of separate subsections, I will briefly discuss crossing and nesting dependencies, roll-up movement, roll-out movement, remnant movement, RNR without coordination, head movement, and the issue of phases.

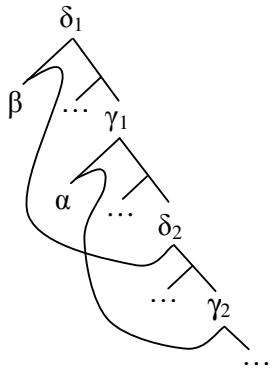
3.5.1. Multiple Instances of Internal Remerge

Internal remerge of more than one constituent is possible. Merge itself facilitates both crossing and nesting movement configurations. Let us investigate if these are generally spelled out correctly according to the present linearization proposal. Two relevant structures are depicted in (57a–b). In both cases, α and β have been internally remerged. Graphically, I positioned them where they ought to be spelled out.

(57) a.



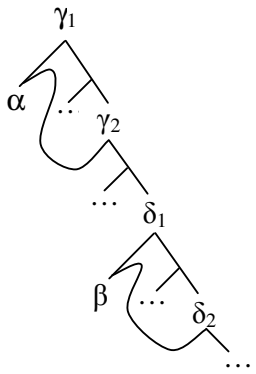
b.



A brief look at (56) will reveal that these structures present no particular problem. Both constituents are linearized the first time they are encountered

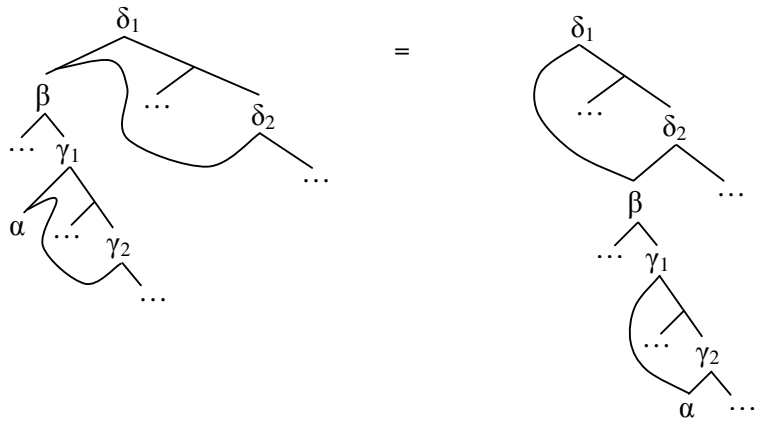
because then the current parent dominates the other parent (which is not yet traversed). Notice that α and β are not related by dominance (neither includes the other). In this respect, (57) is no different from (58), where the two instances of remerge do not interfere in any sense:

(58)



More interesting cases arise if α is included in β. First, consider movement within a moved constituent, as is depicted in two synonymous ways in (59). In a traditional notation, this would correspond to [$_{\delta}$ [$_{\beta}$ … α … t_{α} …] … t_{β} …]:

(59)

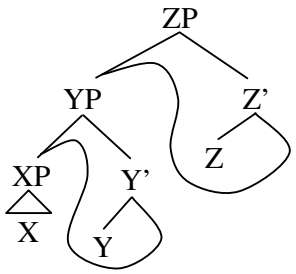


The first occurrence of β is to be linearized. This is performed as in simple movement constructions. Within β, the terminals will be added to the string of words/morphemes. Of these, only the first occurrence of α is to be spelled out; since the parent γ_1 dominates the other parent γ_2, this is indeed the case. The second time β is encountered, when δ_2 is its current parent, the current parent is dominated by the other parent δ_1, so according to (56i), there will be no linearization of β this time. This implies that none of the contents of β will be traversed again, as required.

A special case of iterative internal remerge of the embedding type is so-called roll-up movement; see Barbiers 1995 and Brody 1997, for instance, who use it to mirror the order of PPs across the verbal right sentence bracket, and the order of adjectives across a head noun. An abstract example in traditional notation is [$_{ZP}$ [$_{YP}$ [$_{XP}$ X] [$_{Y'}$ Y t_{XP}]] [$_{Z'}$ Z t_{YP}]]. As a result of these movements, the order of terminals has become the mirror order of how they were first-merged. In

a multidominance graph, the example can be pictured nicely as follows:

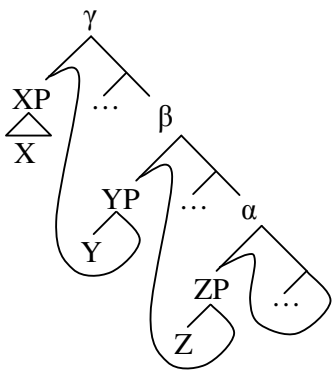
(60)



When traversing this graph, we encounter YP, which has two parents, ZP and Z'. As the first parent dominates the second, only the first occurrence of YP will be linearized. Inside YP, the situation is similar for XP.

In a sense, the structural reverse of roll-up movement is roll-out movement: after internal remerge of a complex constituent, a term of this constituent is remerged even higher, and so on. Abstractly, it looks like (61), where the required order of terminals is /X Y Z/, again a reversal of the order in which the heads were first-merged:

(61)



Here, we recognize a violation of the Freezing principle (Wexler & Culicover 1980). However, one should ask whether it is the task of the phonological interface to exclude such constructions. The answer in this article is negative (recall the discussion in sections 3.1. and 3.2.). If there need to be syntactic constraints concerning sub-extraction, so be it, but that is of no direct concern to the linearization procedure in principle. Apart from that, there are many documented exceptions to the Freezing principle, such as Dutch/German *wat voor/was für*-splits, *wh*-movement from a scrambled constituent, or 'smuggling' in English (for discussion, see Corver 1990, Müller 1998, and Collins 2005, among others). An interesting example of what could in fact be a double violation in German is taken from Ott (2009), who discusses a kind of 'multiple NP split' in detail (proposing an alternative solution in terms of scattered deletion, which goes back to Fanselow & Ćavar 2002). The final trace in (62) is not Ott's but mine, in accordance with a general Head–Comp approach.

(62) *German*

Bücher$_i$	wurden	[so	richtig	gute	$t_i]_k$	in diesem	Jahr	nur	wenige
books	*were*	*PRT*	*really*	*good*		*in this*	*year*	*only*	*few*
$t_k]_n$	rezensiert	t_n.							
	reviewed								

'As for books, only few really good ones have been reviewed this year.'

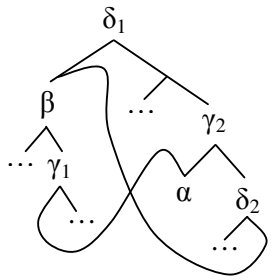
Returning to the abstract structure in (61), we note that XP has two parents: γ and YP. Since YP is dominated by γ, XP (hence X) is spelled out when it is first encountered, that is, if γ is the current parent. In turn, YP has two parents, β and ZP, where β dominates ZP; therefore YP is linearized in the highest position as well. When traversing YP, we spell out Y, and encounter XP for the second time, but here XP cannot be linearized because the current parent, YP is included by another parent, namely γ. And so on. I conclude that the linearization of both roll-up and roll-out movement is correctly performed by the conditioned traversal proposed in (56).

Finally, let us turn to remnant movement, which is perhaps the most complicated of all. An abstract illustration in traditional notation is (63), where α is originally a term of β. A well-known concrete example involves topicalization of a remnant VP after movement of an object to the middle field (for discussion, see den Besten & Webelhuth 1990, Koopman & Szabolcsi 2000, and Müller 2001, for instance).

(63) (…) [$_\beta$ … t_α …] … α_i … t_β (…)

A corresponding multidominance structure is (64), where α has parents γ_1 and γ_2, and β parents δ_1 and δ_2. Concretely, α could be an object, and β a verb phrase.

(64)

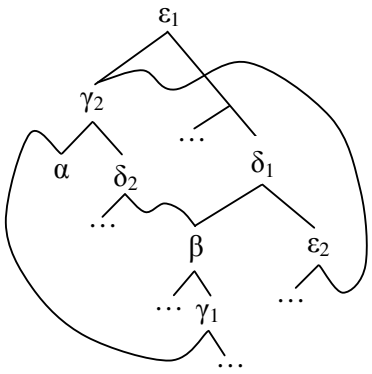


When scanning this structure, we first encounter β via δ_1. As in regular movement constructions, β is linearized in the highest position, since δ_1 dominates the other parent δ_2. Within β, we encounter α via γ_1. Since γ_1 does not dominate α's other parent γ_2, and not all parents have been traversed yet, the linearization of α is omitted here. Later, when α is encountered the second time and γ_2 is the current parent, the remnant α will be linearized because γ_2 is not dominated by the other parent and all parents have been traversed by that time. In retrospect, what makes remnant movement special is that an instance of internal remerge (here, of α) gets the structural appearance of external remerge because of an additional (internal) remerger of another category (β) that dominates the lower position of α.

As a consequence, only the second occurrence of α will be spelled out.

Needless to say, δ_1 in (64) could be input for further roll-up movements. This poses no particular problem. As a final (theoretical) worst-case scenario, consider iterative remnant movement, which would result from moving γ_2 across δ_1 in (64); this is shown in (65), where γ_2's parents are called ε_1 and ε_2:

(65) $[_{\varepsilon 1} [_{\gamma 2} \alpha_i \dots t_\beta] \dots [_{\delta 1} [_\beta \dots t_\alpha \dots] [_{\varepsilon 2} \dots t_{\gamma 2}]]]$



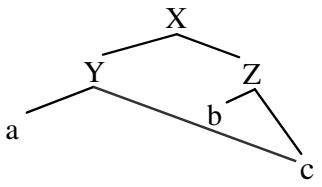
Here, γ_2 will be linearized upon its first encounter (because ε_1 dominates ε_2), and α will be spelled out in the highest position since γ_2 dominates γ_1 (via β). The contents of β are omitted when first encountered via δ_2, and spelled out when δ_1 is the current mother, as required. It is worth noting that there are no actual loops in the multidominance structure in (65), despite its circular appearance at first sight. Therefore, the linearization procedure will have no trouble in ending.

In conclusion, the structures resulting from all possible instances of multiple internal remerge are spelled out correctly, and rather straightforwardly.

3.5.2. Some Issues Concerning External Remerge

The application of external remerge in a derivation may eventually lead to a structural representation that *apparently* involves a violation of the strict cycle. Consider (66), and recall from section 2.3. that such a structure cannot involve *internal* remerge, for the simple reason that Merge functions inherently cyclically (it creates a new root, and does not tamper with the existing structure):

(66)



This abstract representation *can* be derived cyclically by first merging a with c, then externally remerging c by applying Merge (b, c), thereby creating a second root node Z, and finally merging the two temporary roots, Y and Z. We could also start with merging b and c, and then remerge c by Merge (a, c).

The derivation of (66) necessarily involves external remerge, and this has

consequences for the linearization: Applying the conditioned linearization in (56) 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 (66) as being internally remerged with (moved to) the embedded position within Y, as was discussed in section 2.3. Instead, (66) is a type of sharing construction.

A well-known example of sharing is RNR, but section 2.2. suggested that there are many more possibilities. It does not automatically follow from Merge that structures like (66) are only possible in coordinative constructions (that is, where the node joining the double-rooted substructure is a coordination phrase). Of course, we could stipulate such a limitation as a syntactic constraint, but this is not the most interesting way to go. We already briefly touched upon parenthetical-like insertions called amalgams. Here, let me present two examples that fit the pattern in (66). The first involves syntactic subordination of a phrase that contains a shared constituent; see (67), in Dutch, where the shared constituent is printed in italics:

(67) *Dutch*

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

The prepositional phrase *van semantische factoren* 'from semantic factors' is part of the (extended) predicate. This possibility has been noticed before in Huybregts & van Riemsdijk (1985), among others. Examples like (67) give the impression of RNR at the constituent level (note that the adjectives *syntactic* and *semantic* are contrasted). And in fact, verbs like *distinguish*, *compare*, as well as comparative constructions and comitative constructions, are semantically related to coordination. In each case, two or more items with the same selectional properties are used. Syntactically, however, the prepositional connection is subordinative. This leads to the interesting idea that syntactic subordination/ coordination and semantic subordination/coordination are independent of each other. For further discussion, see Postal (1993), Culicover & Jackendoff (1997), van der Heijden (1999), and Lechner (2001).

Another example that is reminiscent of RNR involves a parenthetical-like insertion hierarchically *above* the phrase that contains the shared part; see (68):

(68) *Dutch*

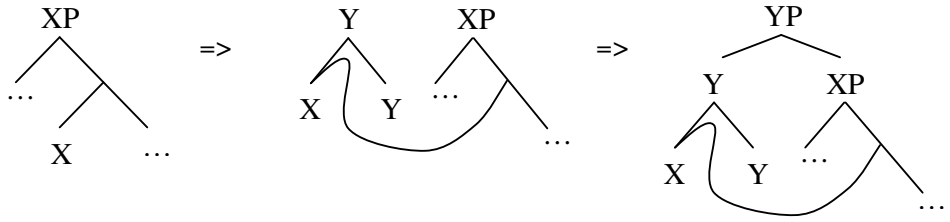
Joop is, ofschoon een schuldbewuste gebruiker van ___,
Joop is although a contrite user of
niettemin principieel gekant tegen *de intracontinentale luchtvaart*.
nevertheless principally opposed against the intracontinental aviation
'Joop is, although a contrite user of ___, nevertheless principally opposed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

Again, there is an implied contrast, and the two prepositions *van* 'of' and *tegen* 'against' must be stressed. The details of the construction need not concern us, here. What is immediately clear is that the structure involves the pattern in (66).

In my analysis, external remerge leads to the pronunciation of the second occurrence of the remerged item — unless it is followed by internal remerge, as in ATB-movement (see sections 2.2. and 3.4.). From this, it can be predicted that instances of *forward* deletion are not sharing constructions involving external remerge, but actual cases of deletion (or ellipsis). Whether this prediction is correct or not must be substantiated by empirical research. What I can say at this point is that indeed the large majority of sharing analyses concern backward deletion constructions, or constructions where there is no overt clue and the analysis could go either way. An important exception, which inspired some others, is Goodall (1987), who also discusses forward gapping from a sharing perspective. However, many differences have been reported between forward and backward deletion in coordination constructions (see Wilder 1997 and de Vries 2005b, for instance), which raises doubts concerning this particular proposal.

Finally, I should add a note on head movement. As the topic is beyond the scope of this article, I will limit myself to a few remarks. A much-discussed problem is that head movement appears to be counter-cyclic (Watanabe 1995). That is, in a structure [$_{YP}$ Y [$_{XP}$… X …]] the head X should not be able to move to Y because Y is embedded in YP. Instead of simply relegating head movement to phonology, many people feel that we should try to find an answer to this problem. One potential solution, first proposed by Bobaljik & Brown (1997), I believe, involves 'sideward movement' — and hence external remerge.[13] The idea is illustrated in (69):

(69)



Merge (X, …) → [X …]
Merge (…, [X …]) → XP
Merge (X, Y) → Y
Merge (Y, XP) → YP

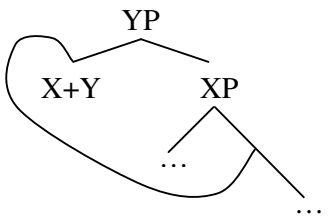
In the crucial step, the head X is externally remerged with an independent item (head adjunction). The combination of both is then merged with XP, such that Y projects.

According to the present linearization proposal, (69) will be treated as a sharing construction. That is, without further assumptions, the second occurrence

[13] Others have argued that head movement does not involve a derived adjunction structure; see, for instance, Koeneman (2000) and Matushansky (2006) for discussion and alternative solutions.

of X (the first-Merge position inside XP) will be pronounced, contrary to fact (at least for 'overt' head movement). Does this mean that we have to abandon the sideward movement approach to head movement? Not necessarily so. Suppose, tentatively, that the final representation in (69) can somehow be transformed into (70) before the actual linearization commences. This would mean that the morphological component is activated at PF. If indeed it recognizes the head incorporation structure created by syntax as a word, it can simplify it into a syntactic unit X+Y (that is, morphological fusion by combining more than one feature bundle under a single category heading). As a consequence, the structure is reinterpreted as a regular movement configuration, with the required linear result.

(70)



Needless to say, this is a non-trivial operation, whose validity needs to be evaluated carefully. I leave the issue open for further research.

3.5.3. What about Phases?

So far, we have discussed the linearization of completed syntactic derivations. However, one may wonder if this approach is compatible with the idea that syntax operates in cycles or phases (see Chomsky 2005 in particular). To the extent that phases can be equated with domains of syntactic locality, the idea is not very controversial. Chomsky also suggests that phases reduce the computational load. Though this is intuitively not implausible, it remains to be shown how it would work out exactly. Importantly, material inside a previous phase of the derivation is no longer accessible for further syntactic computation. Whenever a phase is completed, the contents are transferred to the interfaces. Crucially, it does not follow from this that the phonological component and the semantic component work in phases themselves. It might very well be that the materials transferred by syntax are accumulated until they are complete, and then further processed. Evidence that the phonological and semantic component work in phases (what is more, the *same* phases as syntax does, and in the same direction) can only come from phonology and semantics, respectively, and not from syntax.[14] Furthermore, linearization is probably not a part of phonology proper, but an interface process.

Bearing in mind, then, that the linearization procedure at PF may have access to the complete structure without contradicting the idea of syntactic

[14] Interestingly, there are indications that certain phonological processes work in cycles. To which extent these correspond to syntactic domains is currently debated. For some discussion and further references, see Kratzer & Selkirk (2007).

phases, let us nevertheless examine the possibility that the linearization works in similar phases.

Linearization is essentially a top–down procedure, starting at the root *within each cycle*.[15] By contrast, the syntactic derivation is standardly thought of as bottom-up (here, I cannot discuss alternative proposals, but see, among others, Phillips 2003, Chesi 2007, and Zwart, in press). At the end of each syntactic cycle, the structure is handed over to the PF interface. Suppose that this substructure is linearized right away. When the next cycle is entered, the previous cycle becomes opaque (ideally, in every respect). As long as there is no remerge, this seems possible, indeed. In fact, for each step of Merge (α, β) → γ it is the case that *linearization* (γ) = *linearization* (α) + *linearization* (β). Suppose β is a phase, then the outcome of linearization (a string) is already stored, and the structure of β need not be scanned again. If each projection counts as a phase, that is, if phases are as small as possible, then of course the linearization can be said to behave bottom-up for all practical purposes; see Uriagereka (1999) for a fundamental discussion of 'multiple Spell-Out'.[16, 17]

If, however, remerge is at stake, non-trivial problems arise. The basic trouble, as discussed in section 3.2.2., is that we can no longer rely on Inheritance of precedence.

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 PF at the interface (for the obvious reason that what is in the edge will be used in the higher cycle). It is the complement of the phase head that is transferred ('spelled out'). But then the question arises how the actual spelling out of δ within the lower cycle can be prevented. Its linearization should be

---

15 For LCA-based approaches this may not be entirely true, depending on the implementation. However, recall from sections 3.1 and 3.2.1. that establishing a total order is not the final step in a linearization procedure. In order to map a set of asymmetrical (precedence) relationships between pairs of terminals onto an actual word string, this list of relationships needs to be scanned in a way comparable to what top–down graph scanning amounts to when viewed from the perspective of basic relationships (see section 4).

16 Interestingly, for Uriagereka, in an attempt to adapt the LCA to Bare Phrase Structure, PF linearization crucially makes use of multiple Spell-Out (with minimum-sized phases) in order to cope with complex left branches. This conclusion is almost opposite to the one in this section, for reasons explained immediately below.

17 Another approach making crucial use of phases is Fox & Pesetsky (2005). They claim that the order of elements in one cycle is preserved in the next. Contrary to how it is often referred to, this theory is not an actual linearization *procedure*, as far as I understand it. The central *condition* states that the relative positions of elements must be preserved across cycles. But how these relative positions are actually determined, and how and when it is decided which occurrence of a remerged item is overtly realized are secondary issues from this perspective. I guess any procedure that does the job would be fine, including, perhaps, an adapted version of the present proposal. Notice, however, that the apparent incompatibility of remerge and PF-cycles discussed in the main text constitutes a serious problem for this. For critical comments concerning the idea of order preservation *per se* and the relevant data involved, see Nilsen (2005), among others.

omitted, as Frampton (2004) calls it (meaning that the lower occurrence is not to be pronounced), but the information that δ has another, dominating parent, which would lead to this decision, is no longer present after transfer, during the linearization procedure targeting the structure generated in the lower cycle. Unfortunately, Frampton does not discuss this issue. According to Chomsky (2007: 16), who describes movement in terms of copies instead of remerge, there does not seem to be a problem at all: "[…] all copies are formed by IM [internal merge] at the phase level, hence identifiable for Transfer". To me, honestly, this is more of a mystery than an explanation of how it would work exactly. Perhaps the phase head, which has the higher copy in its specifier position, is able to discern and mark the lower copy as 'lower copy', information that PF can then use after transfer. Clearly, such a move would not be possible in a remerge account, where a lower and higher occurrence involve one and the same item.

I conclude that the introduction of cycles in PF is highly problematic under a remerge account of displacement (disposing of copies or traces), even for standard internal remerge. Not very surprisingly, the situation further deteriorates if external remerge is taken into account:[18]

(D1) External remerge leads to a temporarily multi-rooted structure.

(D2) External remerge shows apparent non-local characteristics, which complicates the need to recover the structural relationship between the relevant parents of a remerged element.

(D3) An externally remerged element must then be first-merged in the cycle where its Spell-Out is to be omitted, but in a bottom-up derivation the relationship between the higher structural parts is not yet fixed, so it cannot be decided which part is, say, the first conjunct, and which the second.

In short, PF linearization must be exempt from the opacity restrictions imposed by phases. This implies that the full syntactic structure must be made accessible to the linearization procedure after the completion of the full sentence. This conclusion is not an argument against phases in syntax: it just means that an essential procedure on the way to phonology does not operate in the same way.

4. A Computational Perspective on Linearization

The linearization of syntactic structure is a procedure performed at PF. Let us examine in some detail what it amounts to. The semi-formal algorithm in (71) combines the basic idea of recursive traversal in (36) with the conditions in (56) for remerged elements. I supplemented it with some comments between braces. Needless to say, if the input is a syntactic object, the output will be a word string.

Lines 1–6 are the initialization and ending. Lines 9–13 are the conditions for

[18] Bachrach & Katzir (2009) 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).

linearization. Here, line 9 refers to single-merged items; lines 10–13 apply to re-merged nodes. Lines 15–21 involve recursive graph scanning. Terminals are pronounced; non-terminals are subject to further analysis, which means going over the entire procedure again at a lower hierarchical level. The elsewhere case in line 22 implies omitting the linearization of the relevant substructure.

The traversal history can be monitored in a very simple way. Each node that is currently inspected is assigned an index; with each subsequent step in the graph, the index is raised by 1. The effect will be that the current node always has the highest index, and the current parent (if relevant) is always the parent with the highest index. Furthermore, each node that has been traversed has an index greater than zero. Thus, it is easy to see that lines 10–13 are equivalent to the conditions in (56).

(71) *Linearization Algorithm for Syntactic Graphs Involving Remerged Nodes*

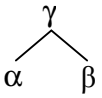
```
algorithm linearization      {performs the conditioned linearization of a binary branching graph
                              with the possibility of multidominance resulting from internal or external remerge}
1  create a new, empty string
2  create a numeral index i with an initial value of 0
3  assign an indexical value of zero to each node      {corresponding to "not yet scanned"}
4  select the root;
5  scangraph                                           {start the scanning procedure with the root}
6  end

procedure scangraph          {conditioned scanning procedure based on recursive traversal}
7  add 1 to the numeral index
8  mark the present object with the numeral index i    {keep track of which node
                                                        is scanned when}
9  if  (the present object has ≤ 1 parents)            {single parent or no parent (for the root)}
10    OR ((there is no parent that dominates the parent with the highest index)
11       AND ((every parent has an index i ≥ 1)        {external remerge: last occurrence}
12             OR (the parent with the highest index dominates
13                   every parent with a zero index)))  {internal remerge: arrival
                                                         from hierarchically highest parent}
14    then                {depth first recursive traversal; add to the string if a terminal is reached}
15       if the present object directly includes members   {is it a non-terminal?}
16          then                                          {traverse both daughters in order of precedence}
17             select the preceding member
18             scangraph                                  {recursive step}
19             select the other member
20             scangraph                                  {recursive step}
21          else add the present object to the string     {"spell out terminal"}
22    else – (do nothing)                                 {"omit linearization"}
23 return
```

The algorithm refers to syntactic structure as if it involves a graph. This is a metaphor, since a graph is only a *representation* of an underlying array of basic syntactic relationships brought about by a series of mergers. Merge creates direct inclusion relationships, and dependency relations between sisters (which is directly mapped onto precedence at PF). We are used to the (ordered) set, bracketed structures, and graph notations, but that implies the following convention, where left–right order, top–down lines, and subscripts are interpreted in a specific way:

(72) *Basic syntactic triad convention*

$<_\gamma \alpha, \beta>$ $=_{def}$

$[_\gamma \alpha\ \beta]$ $=_{def}$



$=_{def}$

} (γ directly includes α) $\wedge$ (γ directly includes β)
$\wedge$ (β is a direct dependent of α /
α directly precedes β)

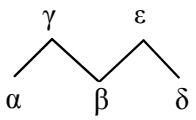
A sequence of mergers produces a list of triads, and hence a list of basic relationships. A full graph or complex set can rather straightforwardly be composed on the basis of this list. Nevertheless, if we want to estimate the computational cost of the algorithm in (71), we have to examine what it does in terms of such a list of local relationships.

In formalized grammars it is often the case that a designated node is defined as the root node. In principle, this is completely unnecessary; because it can be derived which node is the root node by going over the complete list of basic relations (just once). For instance, the list in (73b), which results from the mergers in (73a), can only be interpreted such that ε is the root, as it is the only node that is not included by any other node.

(73) a. Merge (α, β) $\rightarrow$ γ
Merge (δ, γ) $\rightarrow$ ε
b. γ directly includes α, γ directly includes β, α directly precedes β,
ε directly includes δ, ε directly includes γ, δ directly precedes γ

If we find more than one node that satisfies this criterion, the list is not linearizable. For example, the list in (74b), which corresponds to the hydraic structure in (74c), produces two possible roots: γ and ε.

(74) a. Merge (α, β) $\rightarrow$ γ
Merge (β, δ) $\rightarrow$ ε
b. γ directly includes α, γ directly includes β, α directly precedes β,
ε directly includes β, ε directly includes δ, β directly precedes δ
c.



Thus, (74) would crash at the PF interface, contrary to, say, a completed RNR- or ATB-movement construction, in which a uniting instance of Merge has taken place.

When there is a detectable root, the linearization procedure *scangraph* in (71) can be started. In line 9 it is checked if the present object (PO) has one parent or less. This amounts to going over the list and find all items of the type *p directly includes PO*. The parents are temporarily stored in a set $\{p_1, \ldots, p_n\}$, whose cardinality can then be determined.

Line 10 checks if there is no parent that dominates the current parent (which is the parent with the highest index). This is the most costly condition because of the transitive character of dominance. A straightforward way to proceed is as follows. For each parent p_i separately, go over the list and check for items of the form p_i *directly includes* $x_{i1/2}$. Notice that the binary character of Merge makes sure that there will always be two such statements, if applicable. Temporarily store every $x_{i1/2}$ in a set. If this set is not empty, go over the list again and check for items of the form $x_{i1/2}$ *directly includes* y_k. Temporarily store every y_k in a set. If this set is not empty, go over the list again, etc. Finally, check if the current parent is present in the unified set of dominated items $\{x_1, \ldots, x_{n(x)}, y_1, \ldots, y_{n(y)}, \ldots\}$. If this is *not* the case, the condition is fulfilled.

Line 11 checks if all parents have been traversed, which simply involves checking the index of each p_i.

Lines 12–13 check if the current parent dominates every untraversed parent. Thus, we go over the list and check for items of the form $p_{current}$ *directly includes* x_i, and temporarily store every x_i. As before, we go over the list again, now checking for x_i *directly includes* y_k, etc. The set of dominated items can be combined into a unified set. Now, if the subset of parents with a zero index is also a subset of the combined set of items dominated by $p_{current}$, then the condition is fulfilled.

Line 15 checks if the present object is a terminal. This condition is fulfilled if the list of relations contains the items *PO directly includes* x_i. In order to select the preceding or the other child (lines 17 and 19) it is necessary to go over the list again and find the item x_1 *precedes* x_2 or the other way around.

Let us try to estimate the number of steps required to linearize a graph relative to the number of mergers involved in deriving the structure. Note that the number of nodes in a single-rooted graph is directly proportional to the number of mergers (to be precise, two times the number of first-time mergers plus one time the number of remergers plus one), which in turn is one third of the cardinality of the list of basic relations.

After finding the root, the procedure *scangraph* is passed through for each node. In doing so, it is necessary to go over the list of basic relationships a few times (lines 9 and 15–21) for one-parented nodes. Furthermore, because of the possibility of remerged nodes, we must also check the conditions in lines 10–13, which involves going over the list multiple times, thereby looping over the graph depth. Thus, we obtain the formula in (75), which shows polynomial growth:

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

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 (depending on units of computation, etc.). The term $c_1 \cdot M$ involves finding the root; $c_2 \cdot M^2$ is essentially for scanning the entire graph and finding the terminals; $c_3 \cdot M^3$ is the toll for allowing remerge/multidominance.

In fact, 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. With this information available, (75) would reduce to linear growth, if I am not mistaken: $CS = c_0 + c_1 \cdot M$.

5. Conclusion

Complex syntactic objects are derived by the 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 workspace (including syntactic objects that are the output of earlier instances of Merge), and from within complex syntactic objects in the workspace. The last option leads to the effect that an item can be merged more than once, that is, *remerged*. I showed that remerge can be *internal* as well as *external*. Internal remerge, which corresponds to regular movement, is commonly accepted, but external remerge is not. However, both possibilities simply follow from the core system; both involve the basic operation Merge. If one or both are to be excluded, this has to be stipulated explicitly. I argued against such stipulations, not only because minimalist guidelines urge us to, but also since there are sensible interpretations of structures involving external remerge, including several types of amalgams, RNR, and ATB-movement. The sheer variety of relevant construction types can be taken to be a warning to refrain from hastily building in additional constraints on Merge. Furthermore, we must make sure that the linearization procedure at the phonological interface can handle such structures.

Both internal and external remerge can be represented in terms of multidominance. The difference between 'sharing' and '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. This is not to say that there are no differences between constructions involving internal remerge and those involving external remerge. The structural *effect* of remerging with the dominating root or remerging with an external object is quite different. The parents of an internally remerged syntactic object are in a dominance relation, whereas this is not the case for the parents of an externally remerged syntactic object. In the last case, apparent non-local behavior may show up, which can be explained as the possible consequence of a structural 'bypass' between complex substructures that are united at the top. Furthermore, there is no inherent directionality in external remerge. The structural result of first-

merging α with β and remerging it with γ is the same as first-merging α with γ and remerging it with β. Both derivations yield the same configuration with the same structural relationships, and both should be spelled out in the same way. For internal remerge the situation is different: Remerging α with the root is fine, but remerging α with a hierarchically lower position is inherently impossible, since Merge operates strictly cyclically by its very nature, and does not allow 'tampering' of existing relationships.

External remerge creates a temporary multi-rooted structure; this need not be a problem, provided that it is compensated by a uniting instance of Merge at a later stage of the derivation. There are several reasons for such a union; one is that the structure must eventually be linearized, and PF can only interpret a single-rooted object. Thus, even though remerge may lead to unconventional structures, the possibilities are not unconstrained. I argued that there is an additional limitation, dubbed the No proliferation of roots condition, which says that Merge may not create an additional root, which would be counterproductive from a functional perspective. This does not generally exclude external remerge, since the external structure simply keeps having a root of its own, but is does exclude the possibility of remerging an item with another item that is embedded (internally or externally), which would result in undesirable structures. As a consequence, it must always be the case that one of the input elements for Merge is a root at the relevant stage of the derivation.

Evidently, the possibility of remerge complicates the linearization of completed syntactic objects, which must be performed at the PF interface. It is not only the case that remerged items are normally only pronounced once, there also seem to be contradictory linearization demands for internal and external remerge: Informally put, movement is to the left, sharing is to the right. These issues are insufficiently discussed in the literature. All the linearization proposals I am aware of can adequately deal with only a subset of the relevant data. I also argued against LCA-based proposals on principled and practical grounds. Moreover, I showed that the linearization procedure is not likely to operate in cycles. Therefore, I proposed an alternative solution in the form of a conditioned linearization algorithm, which makes use of the different structural configurations created by the two types of remerge. First, it was presented as a graph traversal procedure, in combination with relative structural conditions. I showed that it can handle all kinds of intricate structural patterns involving multiple instances of remerge, including roll-up movement, combinations of sharing and movement, and iterative remnant movement. Subsequently, I showed what this procedure amounts to in terms of a list of basic relationships brought about by a series of mergers, and I estimated the computational load of such a process.

References

Aoun, Joseph & Audrey Li. 2003. *Essays on the Representational and Derivational Nature of Grammar: The Diversity of Wh-Constructions* (Linguistic Inquiry Monographs 40). Cambridge, MA: MIT Press.

Armstrong, David M. 1986. In defense of structural universals. *Australasian Journal of Philosophy* 64, 85–88.

Bachrach, Asaf & Roni Katzir. 2009. Right-node raising and delayed spellout. In Kleanthes K. Grohmann (ed.), *InterPhases: Phase-Theoretic Investigations of Linguistic Interfaces* (Oxford Studies in Theoretical Linguistics 21). Oxford: Oxford University Press.

Barbiers, Sjef. 1995. The syntax of interpretation. Leiden: Universiteit Leiden dissertation.

Barbiers, Sjef, Olaf Koeneman & Marika Lekakou, 2008. Syntactic doubling and the structure of chains. *West Coast Conference on Formal Linguistics (WCCFL)* 26, 77–86.

Barker, Chris & Geoffrey Pullum. 1990. A theory of command relations. *Linguistics and Philosophy* 13, 1–34.

den Besten, Hans & Gert Webelhuth. 1990. Stranding. In Günter Grewendorf & Wolfgang Sternefeld (eds.), *Scrambling and Barriers* (Linguistik Aktuell/ Linguistics Today 5), 77-92. Amsterdam: John Benjamins.

Blevins, James. 1990. Syntactic complexity: Evidence for discontinuity and multidomination. Amherst, MA: University of Massachusetts dissertation.

Bobaljik, Jonathan David. 1995. In terms of merge: Copy and head movement. *MIT Working Papers in Linguistics* 27, 41–64.

Bobaljik, Jonathan David & Samuel Brown. 1997. Interarboreal operations: Head movement and the extension requirement. *Linguistic Inquiry* 28, 345–356.

Brody, Michael. 1997. Mirror theory. *UCL Working Papers in Linguistics* 9, 179-222.

Castillo, Juan Carlos, John Drury & Kleanthes K. Grohmann. In press. Merge over move and the extended projection principle: MOM and the EPP revisited. *Iberia* 1. [Revised and expanded version of Castillo, Juan Carlos, John Drury & Kleanthes K. Grohmann. 1999. Merge over move and the extended projection principle. *University of Maryland Working Papers in Linguistics* 8, 63–103.]

Carnie, Andrew. 2008. *Constituent Structure* (Oxford Surveys in Syntax & Morphology). Oxford: Oxford University Press.

Carstens, Vicky. 2003. Rethinking complementizer agreement: Agree with a Case-checked goal. *Linguistic Inquiry* 34, 393–412.

Chen–Main, Joan. 2006. On the generation and linearization of multi-dominance structures. Baltimore, MD: John Hopkins University dissertation.

Chesi, Cristiano. 2007. An introduction to phase-based minimalist grammars: Why move is top-down and from left-to-right. *CISCL Working Papers on Language and Cognition* 1, 38–75.

Chomsky, Noam. 1981. *Lectures on Government and Binding: The Pisa Lectures* (Studies in Generative Grammar 9). Dordrecht: Foris.

Chomsky, Noam. 1995. *The Minimalist Program* (Current Studies in Linguistics 28). Cambridge, MA: MIT Press.

Chomsky, Noam. 2000. Minimalist inquiries: The framework. In Roger Martin, David Michaels & Juan Uriagereka (eds.), *Step by Step: Essays in Minimalist Syntax in Honor of Howard Lasnik*, 89-155. Cambridge, MA: MIT Press.

Chomsky, Noam. 2001a. Beyond explanatory adequacy. *MIT Occasional Papers in Linguistics* 20.

Chomsky, Noam. 2001. Derivation by phase. In Michael Kenstowicz (ed.), *Ken Hale: A Life in Language* (Current Studies in Linguistics 36), 1-52. Cambridge, MA: MIT Press.

Chomsky, Noam. 2004. Beyond explanatory adequacy. In Adriana Belletti (ed.), *Structures and Beyond: The Cartography of Syntactic Structures*, vol. 3, 104–131. Oxford: Oxford University Press.

Chomsky, Noam. 2005. On phases. Ms., Massachusetts Institute of Technology, Cambridge.

Chomsky, Noam. 2007. Approaching UG from below. In Uli Sauerland & Hans-Martin Gärtner (eds.), *Interfaces + Recursion = Language? Chomsky's Minimalism and the View from Syntax–Semantics*, 1–29. Berlin: Mouton de Gruyter.

Chung, Dae-Ho. 2004. Multiple dominance analysis of right node raising constructions. *Language Research* 40, 791–812.

Citko, Barbara. 2005. On the nature of merge: External merge, internal merge, and parallel merge. *Linguistic Inquiry* 36, 475–496.

Collins, Chris. 2002. Eliminating labels. In Samuel David Epstein & T. Daniel Seely (eds.), *Derivation and Explanation in the Minimalist Program* (Generative Syntax 6), 42–64. Malden, MA: Blackwell.

Collins, Chris. 2005. A smuggling approach to the passive in English. *Syntax* 8, 81–120.

Cormen, Thomas, Charles Leiserson & Ronald Rivest 1990. *Introduction to Algorithms*. Cambridge, MA: MIT Press.

Corver, Norbert. 1990. The syntax of left branch extractions. Tilburg: Katholieke Universiteit Brabant dissertation.

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.

Di Sciullo, Anna Maria & Daniela Isac. 2008. The asymmetry of merge. *Biolinguistics* 2, 260–290.

Engdahl, Elisabet. 1986. *Constituent Questions: The Syntax and Semantics of Questions with Special Reference to Swedish* (Studies in Linguistics and Philosophy 27). Dordrecht: Reidel.

Epstein, Samuel David. 1999. Un-principled syntax and the derivation of syntactic relations. In Samuel David Epstein & Norbert Hornstein (eds.), *Working Minimalism*, 317–345. Cambridge, MA: MIT Press.

Epstein, Samuel David, Erich M. Groat, Ruriko Kawashima & Hisatsugu Kitahara. 1998. *A Derivational Approach to Syntactic Relations.* New York: Oxford University Press.

Fanselow, Gisbert & Damir Ćavar. 2002. Distributed deletion. In Artemis Alexiadou (ed.), *Theoretical Approaches to Universals* (Linguistik Aktuell/ Linguistics Today 49), 66–107. Amsterdam: John Benjamins.

Fiengo, Robert. 1974. Semantic conditions on surface structure. Cambridge, MA: MIT dissertation.

Forrest, Peter. 1986. Neither magic nor mereology: A reply to Lewis. *Australasian Journal of Philosophy* 64, 89–91.

Fortuny, Jordi. 2008. *The Emergence of Order in Syntax* (Linguistik Aktuell/Lingu-

istics Today 119). Amsterdam: John Benjamins.
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, Boston.
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 Studies in Linguistics 46). Cambridge: Cambridge University Press.
Goodman, Nicolas. 1986. Intensions, Church's thesis, and the formalization of mathematics. *Notre Dame Journal of Formal Logic* 28, 473–489.
Gracanin–Yeksek, Martina. 2007. About sharing. Cambridge, MA: MIT dissertation.
Gracanin–Yeksek, Martina. To appear. In Theresa Biberauer & Ian Roberts (eds.), Linearizing Multidominance Structures. *Challenges to Linearization*. Berlin: Mouton de Gruyter.
Grohmann, Kleanthes K. 2003. *Prolific Domains: On the Anti-Locality of Movement Dependencies* (Linguistik Aktuell/Linguistics Today 66). Amsterdam: John Benjamins.
Grootveld, Marjan. 1994. Parsing coordination generatively. Leiden: Universiteit Leiden dissertation.
Grosu, Alexander. 2006. An amalgam and its puzzles. In Hans-Martin Gärtner, Sigrid Beck, Regine Eckhardt, Renate Musan & Barbara Stiebels (eds.), *Between 40 and 60 Puzzles for Krifka.* [http://www.zas.gwz-berlin.de/fileadmin/material/40-60-puzzles-for-krifka/index.html.]
Guimarães, Maximiliano. 2004. Derivation and representation of syntactic amalgams. College Park, MD: University of Maryland dissertation.
Ha, Seungwan. 2008a. On ellipsis features and right node raising. *Conference of the Student Organisation of Linguistics in Europe (ConSOLE)* XV, 67–90.
Ha, Seungwan. 2008b. Ellipsis, right node raising, and across-the-board constructions. Boston, MA: Boston University dissertation.
Haider, Hubert. 2000. OV is more basic than VO. In Peter Svenonius (ed.), *The derivation of VO and OV* (Linguistik Aktuell/Linguistics Today 31), 45–67. Amsterdam: John Benjamins.
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. Nijmegen: Katholieke Universiteit Nijmegen dissertation.
Henderson, Brent. 2007. Matching and raising unified. *Lingua* 117, 202–220.
Heringa, Herman. 2009. A multidominance approach to appositional constructions. Ms., Rijksuniversiteit Groningen.
Heringa, Herman. In progress. Appositional constructions. Groningen: Rijksuniversiteit Groningen dissertation.
Huck, Geoffrey J. & Almerindo E. Ojeda. 1987. Introduction. In Geoffrey J. Huck

& Almerindo E. Ojeda (eds.), *Discontinuous Constituency* (Syntax and Semantics 20), 1–9. New York: Academic Press.
Huybregts, Rini & Henk van Riemsdijk. 1985. Parasitic gaps and ATB. *North East Linguistic Society (NELS)* 15, 168–187.
Jaspers, Dany. 1998. Categories and recursion. *Interface* 12, 81–112.
Johnson, Kyle. 2007. LCA + alignment = RNR. Paper presented at the workshop on *Coordination, Subordination and Ellipsis*, Tübingen. [Eberhard-Karls-Universität Tübingen, 7–8 June 2007.]
Johnson, Kyle. 2009. Why movement? Ms., University of Massachusetts, Amherst.
Karlgren, Hans. 1976. Why trees in syntax? *Studia Linguistica* 30, 1–33.
Karttunen, Lauri & Martin Kay. 1985. Structure sharing with binary trees. *ACL Proceedings* 23, 133–136. [Reprinted in Stuart Shieber, Lauri Karttunen & Fernando C.N. Pereira (eds.), *A Compilation of Papers on Unification-based Grammar Formalisms*, Part II, 5–16. Report no. CSLI-86-48, Center for the Study of Language and Information, Stanford, CA.]
Kayne, Richard S. 1984. *Connectedness and Binary Branching*. Dordrecht: Foris.
Kayne, Richard S. 1994. *The Antisymmetry of Syntax* (Linguistic Inquiry Monographs 25). Cambridge, MA: MIT Press.
Kluck, Marlies. 2007. The perspective of external remerge on right node raising. *Proceedings of CamLing 2007*, 130–137.
Kluck, Marlies. 2008. Intertwined clauses, interacting propositions: A note on the interpretive aspects of sentence amalgamation. *Conference of the Student Organisation of Linguistics in Europe (ConSOLE)* XVI, 77–101.
Kluck, Marlies. 2009. Good neighbors or far friends: Matching and proximity effects in Dutch right node raising. *Groninger Arbeiten zur germanistischen Linguistik* 48, 115–158.
Kluck, Marlies. In progress. The structure and meaning of amalgams. Groningen: Rijksuniversiteit Groningen dissertation.
Kluck, Marlies & Mark de Vries. To appear. Cumulative rightward processes. In Heike Walker, Manfred Sailer & Gert Webelhuth (eds.), *Rightward Movement from a Cross-Linguistic Perspective*. Amsterdam: John Benjamins.
Koeneman, Olaf. 2000. The flexible nature of verb movement. Utrecht: Universiteit Utrecht dissertation.
Koopman, Hilda & Anna Szabolcsi. 2000. *Verbal Complexes* (Current Studies in Linguistics 34). Cambridge, MA: MIT Press.
Koster, Jan. 1999. De primaire structuur. *Tabu* 29, 131–140.
Koster, Jan. 2003. All languages are tense second. In Jan Koster & Henk van Riemsdijk (eds.), *Germania et Alia: A Linguistic Webschrift for Hans den Besten*. [http://www.let.rug.nl/koster/DenBesten/contents.htm.]
Koster, Jan. 2007. Structure preservingness, internal merge, and the strict locality of triads. In Simin Karimi, Vida Samiian & Wendy K. Wilkins (eds.), *Phrasal and Clausal Architecture: Syntactic Derivation and Interpretation* (Linguistik Aktuell/Linguistics Today 101), 188–205. Amsterdam: John Benjamins.
Kratzer, Angelika & Elisabeth Selkirk. 2007. Phase theory and prosodic spellout: The case of verbs. *The Linguistic Review* 24, 93–135.
Kremers, Joost. 2009. Recursive linearisation. *The Linguistic Review* 26, 135–166.

Kural, Murat. 2005. Tree traversal and word order. *Linguistic Inquiry* 36, 367–387.
Kuratowski, Kazimierz. 1921. Sur la notion de l'ordre dans la théorie des ensembles. *Fundamenta Mathematicae* 2, 161–171. [Reprinted in Kuratowski, Kazimierz. 1988. *Selected Papers*. Warszawa: Polish Scientific Publishers.]
Lakoff, George. 1974. Syntactic amalgams. *Proceedings of the Chicago Linguistic Society (CLS)* 10, 321–344.
Langendoen, Terence. 2003. Merge. In Andrew Carnie, Heidi Harley & MaryAnn Willie (eds.), *Formal Approaches to Function in Grammar: In Honor of Eloise Jelinek* (Linguistik Aktuell/Linguistics Today 62), 307–318. Amsterdam: John Benjamins.
Lechner, Winfried. 2001. Reduced and phrasal comparatives. *Natural Language and Linguistic Theory* 19, 683–735.
Matushansky, Ora. 2006. Head movement in linguistic theory. *Linguistic Inquiry* 37, 69–109.
Matushansky, Ora. 2008. A case study of predication. In Franc Marušič & Rok Žaucer (eds.), *Studies in Formal Slavic Linguistics: Contributions from Formal Description of Slavic Languages 6.5*, 213–239. Frankfurt am Main: Peter Lang.
Mayr, Clemens & Viola Schmitt. 2009. Order and the coordinate structure constraint. Ms., Harvard University, Cambridge, MA & Universität Wien.
McCawley James. 1987. Some additional evidence for discontinuity. In Geoffrey J. Huck & Almerindo E. Ojeda (eds.), *Discontinuous Constituency* (Syntax and Semantics 20), 185–200. New York: Academic Press.
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.
Meinunger, André. 2008. Complex numerals as grafts. Paper presented at the *Third Brussels Conference on Generative Linguistics (BCGL 3)*, Brussels. [Hogeschool-Universiteit Brussel, 21–23 May 2008.]
Moltmann, Friederike. 1992. Coordination and comparatives. Cambridge, MA, MIT dissertation.
Mu'adz, Husni. 1991. Coordinate structures: A planar representation. Tucson, AZ: University of Arizona dissertation.
Müller, Gereon. 1998. *Incomplete Category Fronting: A Derivational Approach to Remnant Movement in German* (Studies in Natural Language and Linguistic Theory 42). Dordrecht: Kluwer.
Müller, Gereon. 2001. Two types of remnant movement. In Artemis Alexiadou, Elena Anagnostopoulou, Sjef Barbiers & Hans-Martin Gärtner (eds.), *Dimensions of Movement: From Features to Remnants* (Linguistik Aktuell/Linguistics Today 48), 209–241. Amsterdam: John Benjamins.
Neijt, Anneke. 1979. *Gapping: A Contribution to Sentence Grammar* (Studies in Generative Grammar 7). 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.
Nunes, Jairo. 2004. *Linearization of Chains and Sideward Movement* (Linguistic Inquiry Monograph 43). Cambridge, MA: MIT Press.

Ojeda, Almerindo. 1987. Discontinuity, multidominance, and unbounded dependency in generalized phrase structure grammar: Some preliminaries. In Geoffrey J. Huck & Almerindo E. Ojeda (eds.), *Discontinuous Constituency* (Syntax and Semantics 20), 257–282. New York: Academic Press.

Ott, Dennis. 2009. Multiple NP split: A distributed deletion analysis. *Groninger Arbeiten zur germanistischen Linguistik* 48, 65–80.

Phillips, Colin. 2003. Linear order and constituency. *Linguistic Inquiry* 34, 37–90.

Postal, Paul M. 1993. Parasitic gaps and the across-the-board phenomenon. *Lingu-istic Inquiry* 24, 735–754.

Postal, Paul. 1998. *Three Investigations of Extraction* (Current Studies in Linguistics 29). Cambridge, MA: MIT Press.

Quine, W.V.O. 1945. On ordered pairs. *The Journal of Symbolic Logic* 10, 95–96.

Richards, Norvin. 1999. Featural cyclicity and the ordering of multiple specifiers. In Samuel David Epstein & Norbert Hornstein (eds.) *Working Minimalism*, 127–158. Cambridge, MA: MIT Press.

van Riemsdijk, Henk. 1998. Trees and scions – science and trees. In *Chomsky 70th Birthday Celebration Fest-Web-Page*. Retrieved from http://cognet.mit.edu/Books/celebration/essays/riemsdyk.html.

van Riemsdijk, Henk. 2001a. Wh-prefixes: The case of *wäisch* in Swiss German. In Chris Schaner–Wolles, John Rennison & Friedrich Neubarth (eds.), *Naturally! Linguistic Studies in Honour of Wolfgang Ulrich Dressler, Presented on the Occasion of his 60th Birthday*, 423–431. Torina: Rosenberg & Sellier.

van Riemsdijk, Henk. 2001b. A far from simple matter: Syntactic reflexes of syntax–pragmatics misalignments. In István Kenesei & Robert Harnish, (eds.), *Perspectives on Semantics, Pragmatics, and Discourse: A Festschrift for Ferenc Kiefer* (Pragmatics & Beyond New Series 90), 21–41. Amsterdam: John Benjamins.

van Riemsdijk, Henk. 2006a. Grafts follow from merge. In Mara Frascarelli (ed.), *Phases of Interpretation* (Studies in Generative Grammar 91), 17–44. Berlin: Mouton de Gruyter.

van Riemsdijk, Henk. 2006b. Free relatives. In Martin Everaert & Henk van Riemsdijk (eds.), *The Blackwell Companion to Syntax*, vol. II, 338-382. Oxford: Blackwell.

Ross, John R. 1967. Constraints on variables in syntax. Cambridge, MA: MIT dissertation. [Published revised as *Infinite Syntax!* (Language and Being), Norwood, NJ: Ablex, 1986.]

Sabbagh, Joseph. 2007. Ordering and linearizing rightward movement. *Natural Language and Linguistic Theory* 25, 349–401.

Sag, Ivan & Janet Fodor. 1994. Extraction without traces. *West Coast Conference on Formal Linguistics (WCCFL)* 13, 365–384.

Sampson, Geoffrey. 1975. The single mother condition. *Journal of Linguistics* 11, 1–11.

Schippers, Ankelien. 2008. Partial wh-movement and wh-copying in Dutch: Evidence for an indirect dependency approach. Ms., University of Groningen.

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. Geneva: Université de Genève Dissertation.
te Velde, John. 1997. Deriving conjoined XPs: A minimal deletion approach. In Werner Abraham & Elly van Gelderen (eds.), *German: Syntactic Problems – Problematic Syntax* (Linguistische Arbeiten 374), 231–259. Tübingen: Niemeyer.
Tsubomoto, Atsurô & John Whitman. 2000. A type of head-in-situ construction in English. *Linguistic Inquiry* 31, 176–182.
Uriagereka, Juan. 1999. Multiple spell-out. In Samuel David Epstein & Norbert Hornstein (eds.), *Working Minimalism*, 251–282. Cambridge, MA: MIT Press.
de Vries, Gertrud. 1992. On coordination and ellipsis. Tilburg: Katholieke Universiteit Brabant dissertation.
de Vries, Mark. 2005a. Coordination and syntactic hierarchy. *Studia Linguistica* 59, 83–105.
de Vries, Mark. 2005b. Ellipsis in nevenschikking: Voorwaarts deleren, maar achterwaarts delen. *Tabu* 34, 13–46.
de Vries, Mark. 2005c. Merge: Properties and boundary conditions. *Linguistics in the Netherlands* 22, 219–230.
de Vries, Mark. 2007. Invisible constituents? Parentheses as b-merged adverbial phrases. In Nicole Dehé & Yordanka Kavalova (eds.), *Parentheticals* (Linguistik Aktuell/Linguistics Today 106), 203–234. Amsterdam: John Benjamins.
de Vries, Mark. 2009a. Asymmetric merge and parataxis. *Canadian Journal of Linguistics* 53, 355–386.
de Vries, Mark. 2009b. Unconventional mergers. Ms., University of Groningen.
Watanabe, Akira. 1995. Conceptual basis of cyclicity. *MIT Working Papers in Linguistics* 27, 269–291.
Wexler, Kenneth & Peter Culicover. 1980. *Formal Principles of Language Acquisition*. Cambridge, MA: MIT Press.
Wiener, Norbert. 1914. A simplification of the logic of relations. *Proceedings of the Cambridge Philosophical Society* 17, 387–390. [Reprinted in van Heijenoort, Jean (ed.). 1967. *From Frege to Gödel: A Source Book in Mathematical Logic, 1879–1931*. Cambridge, MA: Harvard University Press.]
Wilder, Chris. 1997. Some properties of ellipsis in coordination. In Artemis Alexiadou & Alan Hall (eds.), *Studies on Universal Grammar and Typological Variation* (Linguistik Aktuell/Linguistics Today 13), 59–107. Amsterdam: John Benjamins.
Wilder, Chris. 1999. Right node raising and the LCA. *West Coast Conference on For-mal Linguistics (WCCFL)* 18, 586–598.
Wilder, Chris. 2008. Shared constituents and linearization. In Kyle Johnson (ed.), *Topics in Ellipsis*, 229–258. Cambridge: Cambridge University 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*, Urayasu. [Meikai University, 2–6 September 2002.]

Yasui, Miyoko. 2004. Syntactic structure without projection Labels. Ms., Dokkyo University.
Zhang, Niina. 2004. Move is remerge. *Language and Linguistics* 5, 189–209.
Zwart, C. Jan-Wouter. 1994. Dutch is head initial. *The Linguistic Review* 11, 377–406.
Zwart, C. Jan-Wouter. 2004. Een dynamische structuur van de Nederlandse zin. Deel 1 en 2'. *Tabu* 33, 55–71 and 151–172.
Zwart, C. Jan-Wouter. 2005. Some notes on coordination in head-final languages. *Linguistics in the Netherlands* 22, 231–242.
Zwart, C. Jan-Wouter. 2006a. Local agreement. In Cedric Boeckx (ed.), *Agreement Systems* (Linguistik Aktuell/Linguistics Today 92), 317–339. Amsterdam: John Benjamins.
Zwart, C. Jan-Wouter. 2006b. Over het enten van interpolaties. *Tabu* 35, 163–180.
Zwart, C. Jan-Wouter. In press. Prospects for top-down derivation. *Catalan Journal of Linguistics* 8.

Mark de Vries
Rijksuniversiteit Groningen
Department of Linguistics
P.O. Box 716
9700 AS Groningen
The Netherlands
Mark.de.Vries@rug.nl