CHAPTER 2

THE FRAMEWORK

1 Introduction

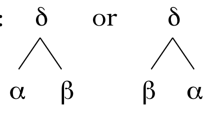
This dissertation adopts a generative perspective on language and assumes a derivational model of the grammar. In particular, it is to be situated in the Minimalist Program (Chomsky 1993, 1995, 2000, 2001). In this modular view of the language faculty, syntactic structures are derived in the computational system C_{HL} via the primitive, recursive structure-building operation Merge. The output of the syntactic computation is sent off to the semantic and phonological component (also named the LF- and PF-interface, respectively). This dissertation focuses primarily on the transfer (Spell-Out) of the syntactic object to PF for pronunciation. More specifically, it takes a closer look at how (multidominant) syntactic structures are linearized into a string. To this end, a cyclic view of the syntax-to-PF mapping and linearization is adopted. The derivation only sends subparts to PF (to be precise, phasal complements and complex left branches). Crucially, it is argued that a fixed linear order once established cannot be changed later on in the derivation. In section 2 of this chapter, I argue that a syntactic object can be 'remerged', which results in this object having two mothers. That is, multidominant phrase markers exist. In section 3, I discuss the specifics of how the linearization algorithm produces consistent linearization statements for multidominant structures in a cyclic Spell-Out/linearization model of the grammar.

This dissertation crucially also centers on the interaction between multidominant phrase markers, cyclic Spell-Out/linearization and ellipsis. Ellipsis is a PF-phenomenon that involves the non-pronunciation of terminal elements and the deletion of linearization statements. I take ellipsis to take place in the course of the derivation, conforming to the cyclic view of the syntax-to-PF mapping. Section 4 discusses the phenomenon of ellipsis.

2 Merge, remerge, and multidominance

In the Minimalist Program (Chomsky 1993, 1995 *et seq.*), it is often assumed that a syntactic structure is constructed out of a numeration (resource, (sub)array, …) N of terminal elements. On the basis of N, the computational system C_{HL} computes a derivation, which will be handed over to the PF- and LF-components. Chomsky (1995) takes the (bottom-up) construction of phrase markers to arise from the primitive structure building operation Merge.[1] Merge is a simple, recursive, grouping operation. Syntactic derivations start out with a collection of terminals to which Merge iteratively applies, until one single phrase marker is constructed from those terminals. Merge combines two syntactic objects α and β, and yields a new, more complex, syntactic object. This new complex object is a set containing the two elements α and β, i.e. {α, β}.[2] It is important to note that Merge not only applies to terminal elements; it can also apply to a complex syntactic object which is itself the output/result of Merge. That is, Merge is recursive and gives rise to syntactic hierarchy. The definition of Merge is given in the two representations in (1):

(1) a. Merge (α, β): {δ, {α, β}}[3]

b. Merge (α, β): 
(i.e. linear order not determined)

Note for (1) that the only constraint Merge imposes when producing phrase markers is binary branching. Merge does not specify linear order: "Just as the sound-meaning relation in the sign is both universal and arbitrary, being left unspecified by UG

[1] The operation Merge belongs Chomsky's (2005) *first factor*, genetic endowment: Merge is given; it is not acquired. As noted by Krivochen (2011:22), "Merge is an operation that 'comes free', […] (a) it is computationally costless and (ii) it cannot be reduced or decomposed."
It has also been proposed that Merge is feature-driven, like all operations in C_{HL} (cf. Adger 2003; Pesetsky & Torrego 2006; Müller 2011). If this view is adopted, though, Merge does not 'come for free'. A feature on an element that conveys 'I am mergeable' justifies the operation. See also Krivochen (2011) for discussion.

[2] For arguments that not Set Merge (as presented here), but Pair Merge is the basic structure building operation, see Jaspers (1998), Langendoen (2003), Zwart (2009a, 2009b, 2011), and De Belder & van Craenenbroeck (2011). Here, I will stick to Set Merge, which is the standard technical implementation of Merge in present-day minimalism. However, my account is also compatible with an implementation of Merge as Pair Merge. See also Citko (2005:146, fn.2).

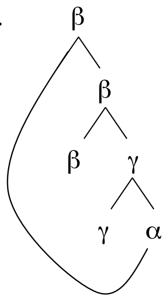
[3] δ is the label of the complex constituent. The value of δ depends on the properties of α and β: either α or β will function as the head of the newly formed constituent. For discussion of projection/labeling, see Chomsky (1995:244ff). See Collins (2002), however, who argues in favor of a label-free syntax (labels are not necessary).

[Universal Grammar], so the relation between the structures (sets of sets of signs) created by Merge and the actual instantiations of those signs, particularly on the PF side, is partially unspecified. [...] In creating sets, Merge does not determine linear order. So UG does not determine the order of elements -- this is subject to variation" (Holmberg & Roberts 2011:2). It is only in the PF-component of the grammar (cf. the linearization algorithm) that syntactic objects get mapped onto linear strings.

Chomsky (2001) distinguishes between two types of Merge: External Merge and Internal Merge. External Merge – called the 'canonical' type of Merge by Citko (2005:475) – takes two distinct, independent root syntactic objects and joins them into one (cf. (1)).[4] Internal Merge (cf. (2)), on the other hand, takes a (possibly complex) subpart of an existing root as one of the two objects and (re)merges it with that root. That is, Internal Merge applies to a syntactic object that has already been merged into one position in the structure, and (re)merges it into a second position. Internal Merge covers the phenomenon of what is traditionally called movement or displacement. As such, movement becomes an epiphenomenon of Merge.

(2) *Internal Merge*: Merge (α, β) when β contains α

e.g.

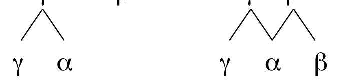


Citko (2005, 2011a) argues that the existence of External and Internal Merge predicts the existence of a third type, which she calls Parallel Merge. Parallel Merge combines the properties of External and Internal Merge. It "is like External Merge in that it involves two distinct rooted objects [...], but it is like Internal Merge in that it combines the two by taking a subpart of one of them" (Citko 2005:476). Similarly, de Vries (2005, 2007, 2009) notes that if 'familiar' Internal Merge is allowed, but the more 'unconventional' Parallel Merge is to be excluded, specific additional

[4] A root syntactic object is a syntactic object that is not dominated by any other syntactic object (see for instance Hornstein et al. 2005:62).

conditions would have to be formulated. In the same vein, van Riemsdijk (2006) argues that if we allow remerge, the application of Parallel Merge can only be excluded by stipulation. Therefore, we expect Parallel Merge to exist in natural language. Parallel Merge is illustrated in (3): α, a subpart of a complex syntactic object γ, is merged with an independent syntactic object β.

(3) *Parallel Merge*



[Citko 2005:476]

Both Internal Merge and Parallel Merge involve 'remerge', i.e. a syntactic object that has been merged before, is merged again. The result is a structure in which a single node (α in (2) and (3)) has two mothers, i.e. in multidominance.[5,6] In the former case, the result is a structure where one of the mothers dominates the other (α dominates β). In the latter case, the result is a multi-rooted structure. That is, in the former case, α and β form a single syntactic object, while they do not in the latter case (they are two independent syntactic objects that share a constituent). As noted by Wilder (2008), and also by Johnson (2009) and de Vries (2007), "two trees 'floating around' as in [(3)] is permissible at non-final stages of the derivation" (Wilder 2008:237). A multi-rooted construct like (3) cannot constitute the final stage of the derivation, as this would violate the Single Root Condition, cf. (4):

(4) *Single Root Condition*
A derivation converges only if (i) the Numeration is exhausted, and (ii) the output consists of a single syntactic object. [Wilder 2008:237]

Hence, a multi-rooted structure like (3) must ultimately be merged into a single syntactic object (for instance by merging γ and β) for reasons of convergence.[7]

Summarizing, there is only one structure-building operation: Merge. *External,*

[5] Multidominance thus abandons the Single Mother Condition (Sampson 1975), see for instance Wilder (1999, 2008) for discussion.

[6] Internal Merge and Parallel Merge are called 'Internal Remerge' and 'External Remerge', respectively, by de Vries (2007, 2009). Gračanin-Yuksek (2007) takes about 'vertical sharing' and 'horizontal sharing', respectively.

[7] For de Vries (2007), a (temporary) multi-rooted structure needs to be merged into a single object before the structure gets linearized. I crucially diverge from this position in this dissertation (see section 3 of this chapter, section 3 and 4 of chapter 3, and section 3 of chapter 5).

Internal, and *Parallel Merge* are just labels referring to Merge selecting different input objects.[8,9,10]

(5)	Merge (α, β) $\rightarrow$ γ constitutes		
	a. *External Merge*	iff	α and β are independent roots before merger
	b. *Internal Merge*	iff	β is a root and α is included in β (or the other way around) before merger
	c. *Parallel Merge*	iff	β is included in some root δ, and α is an independent root (or the other way around) before merger

The next section deals with linearization of multidominant phrase markers in a cyclic Spell-Out/linearization model of the grammar.

3 Linearization and Order Preservation

In the Minimalist Program (Chomsky 1993, 1995 *et seq.*), the syntactic objects built by the computational system (narrow syntax, C_{HL}) are handed over (spelled out, transferred) to the phonological component (PF) for pronunciation.[11] A crucial requirement – "following by 'conceptual necessity' from the legibility conditions imposed at the PF interface if language is to be usable at all" (Richards 2004:10) – is that the terminals of a phrase marker are to be assigned a linear ordering (and

[8] The summarizing overview in (5) is based on de Vries (2007:4).

[9] Apart from the Internal Merge theory of movement (see for instance Epstein et al. 1998 and Gärtner 1999, 2002), multidominant phrase markers have been used to account for various phenomena (such as right node raising, across-the-board WH-questions, coordinated WH-constructions, standard and transparent free relatives, parasitic gaps, parentheticals, sentence amalgamation, etc.). See de Vries (2007, 2009) and Citko (2011a) for an extensive overview.

[10] According to Citko (2005), there is another logical possibility: Parallel Merge that targets subparts of two distinct objects. She does "not see any conceptual reasons to exclude this possibility" (Citko 2005:146, fn.2). See also van Riemsdijk (2006). De Vries (2007), on the other hand, claims that if "α and β are selected as input for Merge, then α or β (or both) must be a root" (his *Root Condition,* de Vries 2007:11), thus excluding this option. As the cases of remerge discussed in this dissertation always involve a root syntactic object, this debate is not my primary concern. For the purposes of this dissertation, I therefore disregard this fourth option.

[11] I gloss over the issue whether the same principles and requirements apply to sign languages. Boeckx (2008:66, fn.2) notes that "the characterization of the syntax-PF interface in sign language studies [...] appears to be isomorphic with the one for spoken languages".

directionality). The question then arises how syntactic structure is mapped onto a linear order at the syntax-PF interface to obtain a legible, i.e. pronounceable, PF-representation. In this subsection, I start out from Kayne's (1994) Linear Correspondence Axiom and then follow Johnson's (2007) reinterpretation of the linearization algorithm to allow for multidominant structures.[12] I take the syntax-to-PF mapping and the linearization algorithm to apply cyclically, as the derivation sends the relevant subparts (phasal complements and complex left branches) to PF (cf. Epstein et al. 1998; Uriagereka 1999; Chomsky 2000 *et seq.*; Epstein & Seely 2002; Fox & Pesetsky 2003, 2004a,b, 2007; Sabbagh 2007). Crucially, linear 'shape' is to be preserved across a derivation: a linearization once fixed cannot be altered later on (cf. Fox & Pesetsky 2003, 2004a,b, 2007; Richards 2004; Johnson 2007; Sabbagh 2007; Engels 2011).

3.1 Kayne's (1994) Linear Correspondence Axiom

As noted by Uriagereka (1998, 1999), the objects created by Merge in narrow syntax are (at least) two-dimensional, whereas speech is one-dimensional. The two-dimensional trees sent to PF (and later to the perceptual-articulatory system) must therefore be mapped onto a one-dimensional phonological representation: they must be given a linear ordering.

In his 1994 monograph, Kayne argues that hierarchical phrase structure completely determines the linear order in which terminal elements (words) are pronounced. His theory is based on the notion of asymmetric c-command (Kayne 1994:4), the definition of which is given in (6).

(6) α asymmetrically c-commands β iff
α c-commands β, and β does not c-command α.

[12] I will not go into other proposals dealing with linearization of multidominant structures, as this would take me too far afield. As Citko (2011a) points out, there are basically four ways to resolve the issue of linearizing multidominant phrase markers: (i) abandon multidominance since it violates the LCA, (ii) abandon the LCA since it disallows multidominance, (iii) modify multidominant structures to make them compatible with the LCA, (iv) modify the LCA to make it compatible with multidominance. Johnson's (2007) proposal can be classified under solution (iv). For alternative proposals, see among others Citko (2005, 2011b), Bachrach & Katzir (2006), Fox & Pesetsky (2007), Gračanin-Yuksek (2007), de Vries (2007, 2009), Wilder (2008). See Citko (2011a) for an overview.

According to Kayne, linearization is sensitive to the asymmetric c-command relation. He proposes the *Linear Correspondence Axiom* (LCA), which maps asymmetric c-command onto a linear ordering of terminals (cf. Kayne 1994:5-6):

(7) *Linear Correspondence Axiom*
d(A) is the linear ordering of T, where
(i) A is the set of all ordered pairs of non-terminals ⟨X, Y⟩ in a given phrase marker P, such that X asymmetrically c-commands Y, and
(ii) T is the set of terminals in P
(iii) d is the non-terminal-to-terminal dominance relation[13]

More specifically, Kayne relates asymmetric c-command to precedence:

(8) Let X, Y be non-terminals and *x*, *y* terminals such that X dominates *x* and Y dominates *y*. Then if X asymmetrically c-commands Y, *x* precedes *y*.
[Kayne 1994:33]

Following (8), if the non-terminals X and Y (the former asymmetrically c-commanding the latter) contain more than one terminal, every terminal in X will precede every terminal in Y.

Note that Kayne's LCA is a formal constraint on the shape of phrase markers, i.e. a property of narrow syntax. Later proposals have limited the place of the LCA in the grammar. The LCA is recast as a PF-mapping strategy: it is a principle of the phonological component, operative only after Spell-Out, because of PF-demands (cf. Chomsky 1995:Ch.4; Uriagereka 1999; Richards 2004). For Kayne, a non-linearizable phrase marker is ill-formed, whereas for the other authors mentioned a non-linearizable phrase marker is ill-formed *only at PF*. Notions of linear ordering play no role in the narrow syntactic component of C_{HL}. I adopt the interpretation of LCA as an interface condition on PF representations.

As noted by Kayne (1994:4), a linear ordering has three defining properties, which he expresses as well-formedness conditions on his LCA. An ordering of

[13] The mapping of asymmetric c-command to linear order is mediated by the concept of *image*, cf. (i). The result of this is that the set of terminals that is the image of one non-terminal, X, is linearized with respect to the set of terminals in the image of another non-terminal, Y.

(i) a. d(X), the image of a non-terminal X, is the set of all terminals dominated by X.
b. d(⟨X, Y⟩), the image of an ordered pair of non-terminals ⟨X,Y⟩, is the set of all ordered pairs of terminals d(X) × d(Y).

terminal elements in a phrase marker is well-formed (linear) if and only if the conditions in (9) are met, where '<' stands for 'precedes':[14]

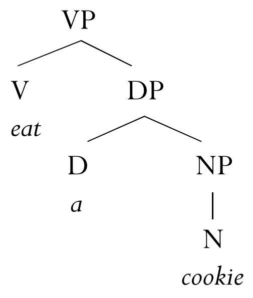
(9) *Well-formedness Conditions on Linearization*

For every distinct terminal x, y, and z in a phrase marker P,

a.	either $x < y$ or $y < x$	⇨	TOTALITY
b.	not ($x < y$ and $y < x$)	⇨	ANTISYMMETRY
c.	if $x < y$ and $y < z$, then $x < z$	⇨	TRANSITIVITY

To see how the LCA system works in practice, consider the following example, with a head (V) and a phrasal complement (DP):

(10)



The values for A and d(A) for the phrase marker in (10) are as follows:

(11) a. A = $\{ \langle V, D\rangle, \langle V, NP\rangle, \langle V, N\rangle, \langle D, N\rangle \}$
b. d(A) = $\{ \langle V, D\rangle, \langle V, N\rangle, \langle D, N\rangle \}$

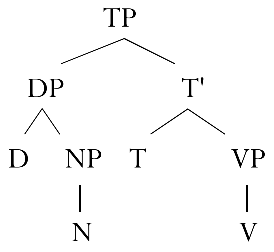
If the ordered pairs in (11)b are taken to represent precedence, this will yield the (expected) linear ordering V < D < N or *eat a cookie*. The ordering in (11)b conforms to the well-formedness conditions on linearization: it is total, antisymmetric and transitive.

As noted by Haumann (2007:45), "[w]hile head-complement relations are straightforwardly captured in terms of asymmetric c-command and the LCA, specifiers and adjuncts [...] appear to fall outside the system." The linearization of

[14] Wilder (2008) adds a fourth well-formedness condition, Irreflexivity (cf. also Stabler 1997 and Nunes & Uriagereka 2000). Irreflexivity requires that for every x and y, if $x < y$, it must be the case that $x \neq y$ (or, in short: not $x < x$) . I follow Wilder (2008:242) that an ordering is linear if and only if it does not violate Totality, Antisymmetry, Transitivity, and Irreflexivity.

subjects (specifiers) and adjuncts (adjoined phrases) – i.e. the linearization of one phrase with respect to another – is indeed problematic. For instance, in (12), the subject DP in [Spec,TP] asymmetrically c-commands the material dominated by T' and T' asymmetrically c-commands the material dominated by the subject DP in [Spec,TP].

(12)



This results in a linear ordering that violates antisymmetry, as shown in the d(A) in (13)b. The d(A) in (13)b contains, for instance, both the statements ⟨T, D⟩ and ⟨D, T⟩, which violates antisymmetry. Thus, the d(A) in (13)b fails to be a linear ordering of the terminals.

(13) a. A = { ⟨D, N⟩, ⟨DP, T⟩, ⟨DP, VP⟩, ⟨DP, V⟩, ⟨T', D⟩, ⟨T', NP⟩, ⟨T', N⟩, ⟨T, V⟩ }

b. d(A) = { ⟨D, N⟩, ⟨D, T⟩, ⟨N, T⟩, ⟨D, V⟩, ⟨N, V⟩, ⟨T, D⟩, ⟨T, N⟩, ⟨V, D⟩, ⟨V, N⟩, ⟨T, V⟩ }

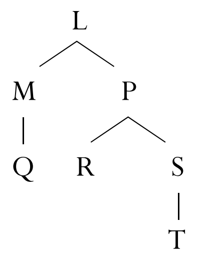
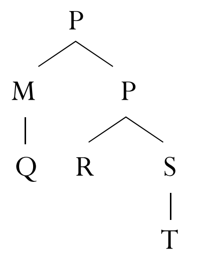
In order to rescue the situation, Kayne (1994:16) has to propose a modification of c-command, complicating the definition by distinguishing between categories and segments (cf. also May 1985, Chomsky 1986). Kayne restricts c-command to categories; a segment cannot enter into a c-command relation.

(14) a. ‘traditional’ c-command:[15]
α c-commands β iff every γ that dominates α also dominates β, and neither α nor β dominates the other

b. Kaynean c-command:
α c-commands β iff α and β are categories and α excludes β and every category that dominates α dominates β[16]

[15] First discussed and defined by Reinhart (1981).

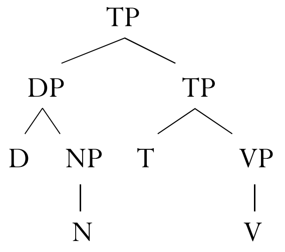
Consider the structures in (15):

(15) a.  b. 

[Kayne 1994:15ff]

In (15)a, P is a category. P asymmetrically c-commands Q and the d(A) of (15)a will contain the pairs ⟨R, Q⟩, ⟨S, Q⟩, and ⟨T, Q⟩. In (15)b, on the other hand, the low P is a segment, not a category. Consequently, P does not asymmetrically c-command Q and the d(A) of (15)b will not contain the pairs ⟨R, Q⟩, ⟨S, Q⟩, and ⟨T, Q⟩.

For the structure in (12)', the result is the A and d(A) in (16), given Kaynean c-command. The linearization in (16)b is total, antisymmetric, and transitive.

(12)' 

(16) a. A = { ⟨D, N⟩, ⟨DP, T⟩, ⟨DP, VP⟩, ⟨DP, V⟩, ⟨T, V⟩ }
b. d(A) = { ⟨D, N⟩, ⟨D, T⟩, ⟨N, T⟩, ⟨D, V⟩, ⟨N, V⟩, ⟨T, V⟩ }

This modification allows Kayne to ensure that the linearizations of subjects and adjoined phrases are LCA-compliant.[17] For more details, I refer the reader to the original 1994 monograph.

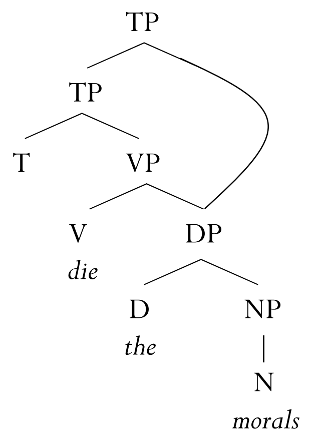
[16] A category α excludes β iff no segment of α dominates β.
α dominates β iff every segment of α contains β. [Kayne 1994:15ff]

[17] Crucial consequences of Kayne's system are (i) specifiers and adjuncts are no longer distinguished and (ii) multiple adjunction is impossible (that is, only one specifier/adjunct per head is allowed).

3.2 Johnson (2007): A modified LCA and multidominance

As noted by Johnson (2007, 2009), Kayne’s LCA (including Kaynean c-command) does not allow for multidominance. The linearization algorithm that converts a (multidominant) syntactic structure into a linear string does not tolerate a terminal that both precedes and follows another terminal. Consider the phrase marker in (17), with the values for A and d(A) in (18). This phrase marker is a simplified multidominant representation illustrating Internal Merge (cf. section 2) of the subject of an unaccusative verb.

(17)



(18) a.

$$A = \left\{ \begin{array}{lllll} \langle D, N\rangle & \langle DP, TP\rangle & \langle T, V\rangle & \langle V, D\rangle & \langle D, N\rangle \\ & \langle DP, T\rangle & \langle T, DP\rangle & \langle V, NP\rangle & \\ & \langle DP, VP\rangle & \langle T, D\rangle & \langle V, N\rangle & \\ & \langle DP, V\rangle & \langle T, NP\rangle & & \\ & \langle DP, DP\rangle & \langle T, N\rangle & & \\ & \langle DP, D\rangle & & & \\ & \langle DP, NP\rangle & & & \\ & \langle DP, N\rangle & & & \end{array} \right\}$$

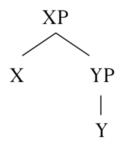
b.

$$d(A) = \left\{\begin{array}{lllll}
\langle D, N\rangle & \langle D, T\rangle & \langle T, V\rangle & \langle V, D\rangle & \langle D, N\rangle \\
 & \langle D, V\rangle & \langle T, D\rangle & \langle V, N\rangle & \\
 & \langle D, D\rangle & \langle T, N\rangle & & \\
 & \langle D, N\rangle & & & \\
 & \langle N, T\rangle & & & \\
 & \langle N, V\rangle & & & \\
 & \langle N, D\rangle & & & \\
 & \langle N, N\rangle & & &
\end{array}\right\}$$

Apart from ⟨D, D⟩ and ⟨N, N⟩ (reflexive statements, cf. footnote 14), the d(A) in (18)b contains several ordered pairs that violate antisymmetry (e.g. ⟨N, V⟩ and ⟨V, N⟩).[18] Kayne's LCA thus bans multidominant structures, as these do not result in well-formed (total, transitive, antisymmetric) linearization statements.[19]

[18] Kayne's well-formedness conditions (cf. (9)) do not exclude reflexive statements. As Johnson (2007:8) puts it, they are "suspicious – but technically allowed". Wilder's (2008) Irreflexivity condition (cf. footnote 14) does prohibit them.

[19] When using (a version of) Kayne's LCA in a framework where Merge is the only structure-building operation, an imperfection arises for a head-complement structure like (i).

(i)


On the one hand, the Merge-based theory does not allow one to start a derivation with a phrase that is made up of one terminal, i.e. Merge simply cannot construct Y(P). On the other hand, the LCA depends on asymmetric c-command to convert hierarchical structure into linear order and can therefore never linearize the merger of two non-branching nodes. Therefore, it is impossible to linearize a head-complement structure with the LCA if the complement contains only one terminal, like in (i). Johnson (2007) and Zwart (2011) argue that the LCA might necessitate positing an empty position. That is, a complement like Y(P) does not consist of a single terminal. Instead, it contains an empty position, which combines with the head Y to form the phrase YP. For Johnson, this empty position is likely to be a phonetically null head. For Zwart, in this case Y merges with the empty set (see also De Belder & van Craenenbroeck 2011).

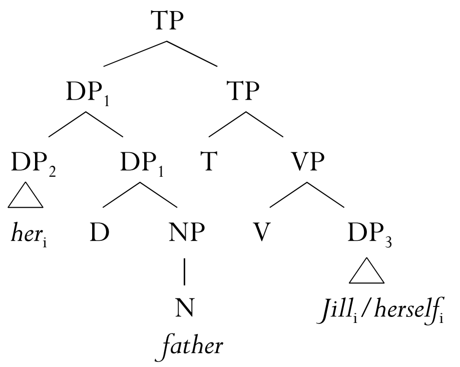
Another solution is proposed by Guimarães (2000), who admits non-branching projections to avoid symmetric c-command. Guimarães suggests that the language faculty allows *Self-Merge,* where a head X is merged with itself. Self-Merge creates the set {α, α}. Guimarães notes that the set {α, α} is identical to the set {α}, following the Extensionality Axiom of Set Theory (Partee et al. 1993). Allowing for non-branching structures avoids LCA violations. For a related proposal, see Kayne (2009).

In this dissertation, I take it to be possible for Merge to form a phrase made up of only one (visible) terminal (represented as in (i)).

Johnson (2007) modifies Kayne's LCA to make it compatible with multidominant phrase markers. Johnson (2007) discusses several problems with the specifics of Kayne's (1994) LCA. For instance, the idea that the linearizations produced by Kayne's linearization algorithm are interpreted as precedence relations is merely a stipulation of the system. Second, a consequence of the modification of c-command in (14) is that it predicts that specifiers c-command out of their phrase. Johnson (2007:9) maintains that "[t]here are many cases where that seems to be wrong". He gives the examples in (19):

(19) [Johnson 2007:9, (27)]
a. Her$_i$ father likes Jill$_i$.
b. * Her$_i$ father likes herself$_i$.

(19)' [cf. Johnson 2007:9, (27)]



In (19)', there are no categories that dominate *her* (DP$_2$). Thus, "every category that dominates *her* also dominates *Jill* [in (19)'a]. Clearly, *her* also excludes *Jill*" (Johnson 2007:9). Given the Kaynean c-command definition in (14)b, it follows that *her* c-commands *Jill.* Johnson (2007:9) concludes that "[t]his should lead to a disjoint reference effect, but there is none. Similarly, *her* will c-command *herself* in [(19)b] and by doing so satisfy the c-command requirement on reflexives. But [(19)'b] is ungrammatical precisely because this requirement is not satisfied."

Here, I do not wish to focus on a detailed discussion of these problems, but on the modifications Johnson (2007) proposes to overcome the conundrums just mentioned and to allow for multidominance. Johnson (2007) preserves the connection between asymmetric c-command and linear order (cf. (7)) as well as Kayne's well-formedness conditions on linearizations (cf. (9)). He rejects, however, the idea that the linear orderings generated by the LCA should be interpreted as precedence and he abandons the modification of c-command in (14). The result of all this is a much freer linearization scheme than Kayne's. The LCA produces sets of ordered pairs whose interpretation is left open: they are ambiguous between a precedence and a subsequence reading. That is, $\langle\alpha, \beta\rangle$ is no longer taken to map onto $\alpha < \beta$, but $\langle\alpha, \beta\rangle = \alpha < \beta$ or $\beta < \alpha$. Thus, the LCA maps an asymmetric c-command relation to a linear statement, but one that is not fully disambiguated. A schematic comparison between Kayne's original LCA and Johnson's modified LCA is given in the table in (20).[20] I adopt Johnson's (2007) proposal. In the table in (20), '<' indicates precedence as usual, '><' is meant to mark 'precedence *or* subsequence'.

(20) *Overview: Kayne's (1994) LCA vs. Johnson's (2007) modified LCA*

constituency	asymmetric c-command	precedence among terminals (Kayne 1994)	precedence among terminals (Johnson 2007)
$[^{X}$ c $[^{Z}$ d e]]	⟨c,d⟩, ⟨c,e⟩	d<e, c<d, c<e	d><e, c><d, c><e
$[^{Y}$ b $[^{X}$ c $[^{Z}$ d e]]]	⟨c,d⟩, ⟨c,e⟩, ⟨b,c⟩, ⟨b,d⟩, ⟨b,e⟩, ⟨b,Z⟩	d<e, c<d, c<e, b<c, b<d, b<e	d><e, c><d, c><e, b><c, b><d, b><e
$[^{W}$a $[^{Y}$ b $[^{X}$ c $[^{Z}$ d e]]]]	⟨c,d⟩, ⟨c,e⟩, ⟨b,c⟩, ⟨b,d⟩, ⟨b,e⟩, ⟨b,Z⟩, ⟨a,b⟩, ⟨a,c⟩, ⟨a,d⟩, ⟨a,e⟩, ⟨a, X⟩, ⟨a, Z⟩	d<e, c<d, c<e, b<c, b<d, b<e, a<b, a<c, a<d, a<e	d><e, c><d, c><e, b><c, b><d, b><e, a><b, a><c, a><d, a><e

Clearly, interpreting all ordered pairs as involving both precedence and subsequence will lead to inconsistent and conflicting linearization statements in d(A). Thus, compared to Kayne's proposal, Johnson's system allows the linearization scheme to generate a far greater number of ordering statements, which possibly violate the well-formedness conditions in (9). Johnson calls this property of the linearization algorithm TOLERANCE (Johnson 2007:14).

Johnson proposes that this is allowed, as long as there is a proper subset that results in a total and consistent linear order. The ordered pairs generated thus need to be disambiguated. Johnson (2007:12) "gives to output constraints the work of interpreting as precedence the ordered pairs that the LCA produces". In short, output constraints disambiguate the ordered pairs, selecting a subset that will result in a total linearization. Johnson proposes that next to Kayne's well-formedness

[20] This schematic representation is based on a similar table in Guimarães (2004:218, (30)).

conditions in (9) – conditions on d(A), which has linearization information about terminals – there are also well-formedness conditions that hold of ordered pairs containing phrasal information. The latter well-formedness conditions are language-specific requirements. Languages make a choice to put heads either at the left or at the right of their phrases (e.g. there is a requirement that "verbs precede their complements"). This is reminiscent of the head parameter of *Government & Binding*, but, as noted by Richards (2004:7), in this case it is a PF ordering strategy rather than a syntactic (phrase-structural) constraint. Another example of a similar condition giving a language its particular word order is "specifiers come initially in their phrases". A subset of the ordered pairs is selected that meets the language-particular requirements. As such, language-specific requirements seem to function as what could be called filters, filtering out a subset of the asymmetric c-command relations generated by C_{HL} and the concomitant ordered pairs that did not get an interpretation in terms of precedence or subsequence.[21,22] The resulting subset is the maximally small subset that will lead to a total linearization. Given that the subset has to yield a complete linearization that meets Kayne's well-formedness constraints, inconsistent statements are jettisoned. From this subset, a d(A) is produced, which has to meet the well-formedness conditions in (9). Concluding, instead of rejecting a linearization on the basis of (violations of) the well-formedness conditions (as in Kayne's original (1994) proposal), subsets are selected that meet language-specific requirements and Kayne's well-formedness conditions, in the end resulting in a total, consistent linearization.[23]

The idea of resolving conflicting ordering statements at PF is not only present in Johnson (2007). Epstein et al. (1998) introduce the *Precedence Resolution Principle (PRP)*, which ignores a subset of c-command relations in the narrow syntax.[24]

[21] Similar ideas can be found in Johnson (2009, 2010a, 2011a). Actually, in his (2007) proposal, Johnson calls the well-formedness conditions on A "alignment constraints". He does not adopt this term in later work, where he focuses on 'language-specific requirements'.

[22] As such, this 'filtering' of asymmetric c-command relations and concomitant ordered pairs is an alternative to Kayne's (1994) proposals for disregarding certain asymmetric c-command relations, such as altering the definition of c-command and waiving the distinction between specifiers and adjuncts.

This is in line with the idea that Merge is free (cf. also chapter 3) and that the only 'filters' for representations are interface conditions. As noted by Krivochen (2011:24), "[t]hese conditions determine the set of what is known as 'convergent derivations', a subset of the possible derivations."

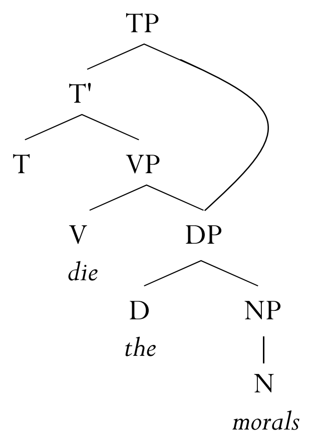
[23] This proposal is illustrated in section 3.4 of this chapter, combined with a cyclic Spell-Out model of the grammar.

[24] Epstein et al. (1998) propose that c-command, rather than asymmetric c-command, translates to precedence at PF (this is also adopted by Richards 2007). In that case, mutual c-command relations result in conflicting ordering statements at PF. I follow Johnson (2007) in adhering to Kayne's original (*continued on the next page*)

According to Epstein et al., the ordering relation in the base must be ignored at PF. Richards (2007) proposes a strong version of the *PRP* that applies throughout the derivation. According to Richards, derivational information is simply deleted and the original order imposed by the *PRP* in the base is preserved (contra Epstein et al.). Thus, although details differ, Johnson's proposal to delete ambiguous information at the interface is comparable to other proposals in the literature.

A consequence of Johnson's (2007) proposal is that the linearization scheme is no longer incompatible with multidominant structures. Let us consider the phrase marker in (21) again. The A in (22) lists all asymmetric c-command relations in (21). Remember that ⟨X, Y⟩ no longer maps onto X < Y.

(21)



(22)

$$A = \left\{ \begin{array}{lllll} \langle D, N\rangle & \langle DP, T\rangle & \langle TP, D\rangle & \langle T, V\rangle & \langle V, D\rangle \\ & \langle DP, VP\rangle & \langle TP, NP\rangle & \langle T, DP\rangle & \langle V, NP\rangle \\ & \langle DP, V\rangle & \langle TP, N\rangle & \langle T, D\rangle & \langle V, N\rangle \\ & \langle DP, DP\rangle & & \langle T, NP\rangle & \\ & \langle DP, D\rangle & & \langle T, N\rangle & \langle D, N\rangle \\ & \langle DP, NP\rangle & & & \\ & \langle DP, N\rangle & & & \end{array} \right\}$$

As proposed by Johnson (2007), the LCA maps a relation of asymmetric c-command

proposal that it is asymmetric c-command translates to linear order (and I therefore do not discuss Epstein et al.'s and Richards' accounts any further here).

onto an ordered pair, the interpretation of which (in terms of precedence or subsequence) is left open. That is, ⟨α, β⟩ does not map onto to α < β: ⟨α, β⟩ = α < β or β < α. The pairs corresponding to the asymmetric c-command relations in A in (22) are disambiguated on the basis of language-particular requirements. For example, English is head-initial and requires heads to precede their complements. This disambiguates an ordered pair like ⟨D, N⟩ as D < N, given that the D-head of DP has to precede its complement. Similarly, specifiers need to precede the other material in their phrase. This disambiguates an ordered pair like ⟨DP, T⟩ as DP < T, with the DP-specifier preceding the head of the phrase. Finally, given that the subset has to yield a complete linearization that meets Kayne's well-formedness constraints, inconsistent pairs such as ⟨DP, DP⟩ are deleted. A maximally small, disambiguated subset of the ordered pairs is selected that meets the language-specific requirements and will lead to a total linearization of all the terminals. "Maximally small" means that redundant statements are thrown out (although these would have no influence on the linearization). The result is (23).[25] On the basis of (23), the linearization in (24) is produced.

(23)

$$A' = \left\{ \begin{array}{lll} D < N & DP < T & T < V \\ & DP < V & \\ & & \end{array} \right\}$$

(24)

$$d(A) = \left\{ \begin{array}{lll} D < N & D < T & T < V \\ & N < T & \\ & D < V & \\ & N < V & \end{array} \right\}$$

The linearization in (24) satisfies the well-formedness constraints: it is total, antisymmetric, and transitive. The structure in (21) is thus spelled out as the

[25] Johnson (2007:13-14) names the set in (23) "A" and calls it "a subset of A that meets the [language-specific] constraints". This is, however, confusing, as A is actually "the set of all ordered pairs of non-terminals ⟨X,Y⟩ in a phrase marker P, such that X c-commands Y" (cf. (7)). That is, A contains information on c-command, not on precedence or subsequence. The set in (23) is therefore not straightforwardly "a subset of A". To avoid confusion, I will name sets such as (23) A'. A' is meant to refer to the maximally small, disambiguated subset of the ordered pairs that correspond to the asymmetric c-command relations in A.

grammatical sentence *the morals die*. As such, Johnson's linearization algorithm allows multidominant structures "by allowing the linearizations to generate inconsistent orderings just so long as there is a proper subset of that linearization that is consistent and total" (Johnson 2007:14).[26]

This subsection presented the basics of Johnson's (2007) proposal for linearizing multidominant phrase markers. The next section introduces cyclic Spell-Out/linearization, which will play a crucial role in this dissertation. As will become clear in section 3.4 of this chapter, some refinements to Johnson's (2007) proposal are required in a system that both allows multi-rooted phrase markers and incorporates cyclic Spell-Out/linearization. The requirement of TOTALITY needs to be rephrased and d(A) is shown to also have the property of TOLERANCE.

3.3 Cyclic linearization and Order Preservation

3.3.1 PHASES AND MULTIPLE SPELL-OUT

In this dissertation, I adopt a derivational system of computation that combines a Chomskyan phase model with Uriagereka's (1999) Multiple-Spell out proposal. I take both phasal domains and complex left branches to undergo Spell-Out, after which they become inaccessible.

Chomsky (2000, 2001 *et seq.*)'s central hypothesis is that syntactic structures are built up one cycle at a time, after which they are spelled out. The rationale behind this is the reduction of computational burden (memory load) via the 'periodic' forgetting of derivational information (cf. Richards 2011:74). The Language Faculty can only process limited amounts of structure at one time and, more specifically, can only hold a limited amount of structure in its 'active memory' (cf. Chomsky 1999:9).

[26] Johnson (2009, 2010a, 2011a) suggests a different solution for the problem posed by linearizations of multidominant phrase markers. He proposes that the linearization algorithm need not produce linearization statements that will violate the LCA to begin with. The linearization algorithm in Johnson (2009, 2010a, 2011a) does not evaluate all of the positions a terminal occupies in the course of the derivation. A terminal that occupies two (or more) positions is linearized in only one spot in the string (this goes back to Nunes' (2004) *Chain Reduction* for the linearization of non-multidominant structures).

This is, however, based on a representational, non-cyclic view of multidominant structures, where all positions of a remerged phrase are visible 'at once'. When cyclic Spell-Out and linearization enter the picture, it is not clear how the linearization algorithm can simply ignore certain positions occupied by a remerged phrase. Johnson (2011a:31) also explicitly rejects the possibility of linearization statements that violate the LCA. In this dissertation, I adopt the specifics of the linearization scheme proposed in Johnson (2007).

Narrow syntax derives small derivational subparts or chunks of structure, called *Phases*.[27] At any one time, the derivation can access only one phase, limiting the computational load in deriving a sentence (cf. Hicks 2009:43). Phases are made up of the phase head, its domain (i.e. its complement), and its edge (i.e. its specifiers/adjuncts). Chomsky's original (2000) proposal is that upon completion of the phase, the phasal domain is sent off (or transferred) to the interfaces and becomes inaccessible to further syntactic operations. The phase head and the phase edge, on the other hand, are not transferred until the next phase is completed: they remain accessible at the next higher phase, thus making the phase edge available as an escape hatch. This ensures that long-distance movement proceeds phase edge by phase edge, i.e. successive-cyclically. This is formally stated by the *Phase-Impenetrability Condition* (PIC) in (25):[28]

(25) *The Phase-Impenetrability Condition (PIC)*

In a phase α with a head H, the domain of H is not accessible to operations outside α, only H and its edge are accessible to such operations.

[Chomsky 2000:108]

This entails that Spell-Out also applies in a cyclic manner: there are multiple, cyclic applications of Spell-Out in the course of the derivation, applying to specific subparts of the syntactic object. Here, I take 'Spell-Out' to be interpreted as the operation that takes a syntactic object and 'hands it over' to the PF component of the grammar, also called 'Transfer to PF', to be distinguished from 'Interpret'/'Transfer to LF' (cf. Chomsky 2004, Lasnik et al. 2005). Spell-Out thus applies only to PF (cf. Nissenbaum 2000, see also Marušič 2005 who distinguishes

[27] There is quite some debate in the literature about which chunks of the derivation constitute phases. In its original conception, only CP and the vP of transitive and unergative verbs are considered phasal (cf. Chomsky 2000:106). Unaccusative and passive vPs, DPs, and PPs have been taken to qualify for phasehood as well (cf. Legate 1998, 2003; Abels 2003; Richards 2004; Svenonius 2004; Chomsky 2005, 2008; Sabbagh 2007). In what follows, I take CP and vP (whether transitive, unergative, unaccusative, or passive) to constitute phases (cf. Chomsky 2000 *et seq.*; Legate 1998, 2003; Richards 2004).

[28] There are two versions of the *Phase-Impenetrability Condition*, the original from Chomsky (2000) (sometimes called PIC1) and a modified version from Chomsky 2001 (PIC2), cf. (i):

(1) *The Phase-Impenetrability Condition* (alternative) [Chomsky 2001:14]

The domain of H is not accessible to operations at ZP [a phase];
only H and its edge are accessible to such operations.

That is, a phasal domain is only transferred (and rendered inaccessible) when the next phase (ZP) has been completed. In this dissertation, I adopt Chomsky's original (2000) proposal that the phasal domain is transferred upon completion of the phase (see also Nissenbaum 2000; Hiraiwa 2002; Hicks 2009; Richards 2011).

non-simultaneous PF and LF phases). I will adopt this terminology, as this dissertation is mostly concerned with the syntax-PF interface. Hence, when I use the term 'Spell-Out' in the following chapters, it is intended to mean 'Transfer to PF'.

Uriagereka (1999) – picking up ideas first expressed in Bresnan (1971) – considers every branching left branch to be targeted by Spell-Out.[29] In his Multiple Spell-Out model, complex (that is, branching) structures in specifier and adjunct position are separate derivational chunks that undergo Spell-Out before merging with the rest of the tree structure.[30] According to Uriagereka, after Spell-Out the specifier or adjunct is flattened into an ordered sequence and reenters the derivation as a "giant lexical compound" (Uriagereka 1999:256). The computation does not recognize it as syntactically complex element, but treats it as an atom, a word, an X^o. Material inside the specifier or adjunct is therefore no longer accessible to syntactic operations. A consequence of Uriagereka's proposal is that it derives the islandhood of subjects and adjuncts (Ross 1967).[31] Although I do not consider complex left branches to be X^o's (cf. also Johnson 2002:4 for difficulties with this specific implementation), I do, adopt the proposal that they get spelled out upon merging in the 'main' structure and are opaque for further syntactic computation.[32] Thus, complex left branches resemble phasal domains in that they are spelled out (i.e. transferred to PF) and become inaccessible for further operations.

For independent support that integrating Chomsky's and Uriagereka's dynamic models is the desired road to take, see for instance Sato (2006, 2009). Sato proposes that Spelled-Out mid-derivational objects are mapped to prosodic domains at the PF interface (which he calls the *Syntax-Prosody Mapping Hypothesis*). According to Sato, if separate derivational chunks reach the interface, this should have repercussions for the domain of prosodic rule application. He argues that a combination of Uriagereka's Multiple Spell-Out hypothesis and Chomsky's Phase theory is required

[29] It has even been suggested (cf. Epstein et al. 1998; Epstein & Seely 2002, 2006) that a derivation is dynamically accessed by the interface at every derivational step ("Spell-Out-as-you-Merge"). Nakamura (2009) holds that any projection can in principle be a phase (cf. also Boeckx & Grohmann 2007). I do not adopt these proposals here.

[30] Here, we abstract away from the questions (i) whether the two phrase markers (the left branch and the clausal spine) are assembled simultaneously in separate derivational spaces or sequentially in the same derivational space, and (ii) whether (and if so, how) the Spelled-Out left branch is renumerated (cf. Uriagereka 1999; Johnson 2002; Postma & Rooryck 2009; Krivochen 2011; De Belder & van Craenenbroeck 2011).

[31] Other authors have adopted the idea that islandhood should be subsumed under phasehood, or, more general, that islands are targeted by Spell-Out and therefore become opaque (Johnson 2002, 2007, 2009; Fox & Pesetsky 2003, 2004a; Sabbagh 2007; Krivochen 2011).

[32] The proposal that a Spelled-Out constituent becomes opaque for further syntactic computations is also adhered to by, for instance, Zwart (2009b) and De Belder & van Craenenbroeck (2011).

for the proper access from syntax to phonology. Sato maintains that combining these dynamic models yields empirical predictions about possible structural domains for prosodic rule application and he shows that these are borne out by a variety of phonological alternations across languages. Sato discusses Taiwanese tone sandhi, French liaison, Gilyak lenition, Kinyamboo high deletion, and Welsh consonant mutation.

First of all, according to Sato, Uriagereka's model predicts that a complex specifier/adjunct configuration forms an independent prosodic domain, while a simplex specifier/adjunct configuration forms a prosodic domain that also includes a head/complement.[33] This prediction is for example borne out by Kinyamboo high deletion (Sato 2006:10-11). Kinyamboo has three surface tones: High (á), Low (a), and Falling (áa). In Kinyamboo, a High tone in a word is deleted when it is immediately followed by another word with high tone (see also Bickmore 1990). This is illustrated in (26).

(26) [Sato 2006:10, (37)]

omuk**a**ma mukázi (cf. omuk**á**ma 'chief' (in isolation))
chief old
'old chief' [Kinyamboo]

Sato shows that when a specifier is complex, its High tone is maintained: in (27)a, the complex subject *abakozi bakúru* 'the mature workers' has a High tone on *bakúru* 'mature'. When a specifier is simplex, on the other hand, its High tone is deleted: in (27)b, the simplex subject *abakozi* 'workers' has lost its High tone. This confirms Sato's prediction based on Uriagereka's model.

(27) *complex vs. simplex subject* [cf. Sato 2006:11, (38)]

a. [abakozi bak**ú**ru] bákajúna (cf. bak**ú**ru 'mature' (in isolation))
workers mature they helped
'The mature workers helped.'

b. [abak**o**zi] bákajúna (cf. abak**ó**zi 'workers' (in isolation))
workers they helped
'The workers helped.' [Kinyamboo]

[33] Uriagereka (1999:262-265) himself also points out that his model receives empirical support from focus spreading, pauses/parenthetical expressions, phonological association of certain function items to the lexical heads and the cliticization of determiners to their preceding heads in Galician.

Sato argues, however, that Uriagereka's model cannot account for all syntax-prosody mappings; Chomsky's Phase theory has to be incorporated as well. One of his arguments is based on Welsh consonant mutation (Sato 2006:11-12). Consonant mutation in Welsh is the phenomenon that an initial consonant of the citation form of a word undergoes the replacements in (28) in certain syntactic configurations. This is shown in (29) for c → g mutation.

(28) [Sato 2006:11, (40)]

p	→	b	b	→	f	m	→	f
t	→	d	d	→	dd	rh	→	r
c	→	g	g	→	NULL	ll	→	l

(29) *Consonant mutation* [cf. Sato 2006:11, (41)]
Gwenlodd y dyn **g**i. (cf. **c**i 'dog (in isolation)')
see-PAST-3S the man dog.
'The man saw a dog.' [Welsh]

Sato argues that CP clauses, but not TP-clauses, constitute barriers for consonant mutation (see also Harlow 1989; Tallerman 1990; Roberts 1997). This is illustrated in (30). The contrast between (30)b and (30)c shows that, while mutation (b → f) can occur in a TP-clause (cf. (30)b), it cannot in a CP-clause (cf. (30)c)): *fod* 'be' is ungrammatical in a CP-clause.

(30) [cf. Sato 2006:12, (42)-(43)-(44)]

a. Dywedodd [$_{NP}$ hi] [$_{CP}$ (y) [$_{IP}$ **bydd** hi'n prynu car newydd]].
say-PAST-3S she COMP will-be-3S she-PROG buy car new
'She said (that) she will be buying a new car.'

b. Dywedodd [$_{NP}$ yr athro] [$_{IP}$ **fod** Gareth wedi colli'r bws].
say-PAST-3S the teacher be Gareth PERF lose-the bus
'The teacher said Gareth had missed the bus.'

c. * Dydy o ddim yn credu [$_{CP}$ y [$_{IP}$ **fod** Gwyn yn dweud y gwir].
NEG-is-3S he NEG PROP believe COMP be Gwyn PROG say the truth
'He doesn't believe that Gwyn is telling the truth.'
[Welsh]

Sato argues that Uriagereka's model cannot answer the question why CPs, but not TPs, block consonant mutation: Uriagereka's model "does not draw any distinction between CP and TP nodes that would be pertinent to Spell-Out" (Sato 2006:12). Sato proposes that incorporating Chomsky's Phase Theory provides the answer. While C is a phase head, T is not, and only the phasal C-head, not the T-head, creates a boundary for consonant mutation in Welsh. According to Sato, the CP-TP distinction in Welsh consonant mutation provides phonological support for the CP phase under the *Syntax-Prosody Mapping Hypothesis*.[34]

Sato concludes that Spelled-Out derivational chunks are mapped onto prosodic domains at PF. Only a combination of both Uriagereka's and Chomsky's dynamic models can account for all empirical observations.

3.3.2 CYCLIC LINEARIZATION AND ORDER PRESERVATION

We have seen in the previous subsection that at specific points in the derivation, particular phrases (specifically, phasal complements and complex left branches) undergo Spell-Out. Cyclic Spell-Out transfers derivational subparts to PF. It is generally assumed that "the linearization of syntactic structures is [...] part of the Spell-out procedure, that is, the mapping to PF" (Fuß 2005:90). It has been proposed that dynamic, cyclic Spell-Out also implies dynamic linearization (cf. Fox & Pesetsky 2003, 2004a,b; Richards 2004; Müller 2005; Sabbagh 2007; Engels 2011). Each spelled-out chunk constitutes a separate linearization domain and the linearization algorithm also applies cyclically. Each application of Spell-Out thus provides a set of linear ordering statements.

Fox & Pesetsky (2003, 2004a,b), Johnson (2002, 2007, 2011a), Sabbagh (2007), and Engels (2011) push the idea of cyclic linearization even further. These authors argue that for each linearization domain, the linearization algorithm fixes a linear order, once and for all. It immutably assigns a position to each terminal element in the relevant phrase marker. Thus, if a terminal α is set to follow another terminal β ($\beta < \alpha$) in some linearization domain, this ordering cannot be changed later on: the final linearization statement cannot be $\alpha < \beta$. Ordering statements are 'carried over' from lower linearization domains to the higher ones. This is called Linearization Preservation or Order Preservation.

[34] Sato (2006, 2009) also refers to Bošković (2001) and Bošković & Lasnik (2003) for additional support: these authors argue that the C-head creates an intonational boundary and blocks PF-affixation/merger. Note that Dobashi (2003) also proposes a hypothesis similar to the *Syntax-Prosody Mapping hypothesis* for phonological phrasing within Chomsky's Phase Theory.

(31) *Order Preservation*
Linear order is fixed once and for all at the end of each linearization domain.
[based on Fox & Pesetsky 2003:2]

3.4 Multidominance, modified LCA, cyclicity: Illustration

As I indicated above (subsection 3.3.1), I adopt a derivational system of computation that combines Uriagereka's Multiple Spell-Out proposal with a Chomskyan phase model. Spell-Out is cyclic and the linearization algorithm also applies cyclically (cf. subsection 3.3.2). I follow the proposal of the authors mentioned in the previous subsection that the application of the linearization algorithm fixes an unchangeable linear order for the terminal elements of the relevant phrase marker (i.e. Order Preservation).

In this subsection, I present Johnson's (2007) analysis of Right Node Raising, which brings together a multidominant structure, Johnson's modified LCA, mid-derivational Spell-Out/linearization and Order Preservation. The case of Right Node Raising therefore nicely illustrates how a multi-rooted structure gets linearized in a cyclic Spell-Out/linearization model.

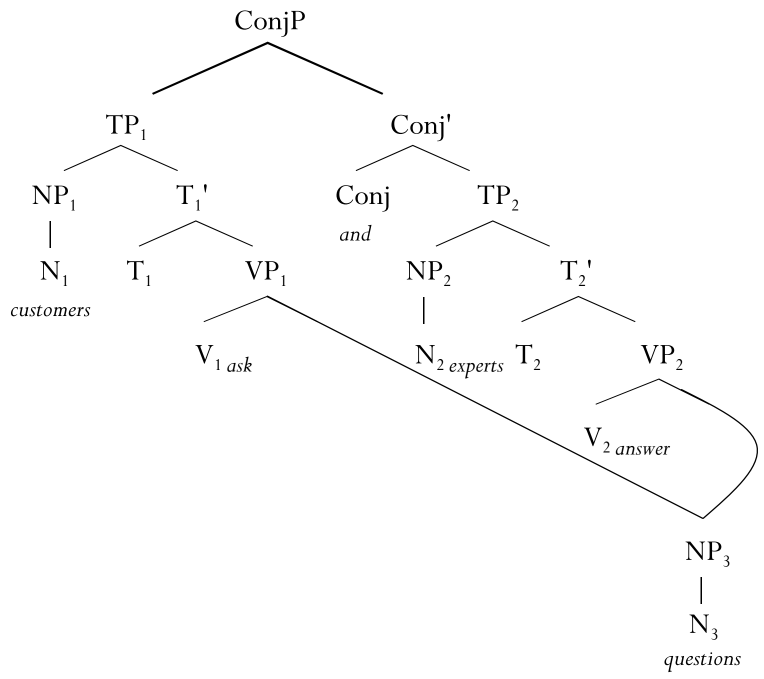
(32) Consumers ask and experts answer questions.

Although the account is not uncontested (cf. Postal 1998; Sabbagh 2007; Ha 2008), Right Node Raising (RNR) examples like (32) have been analyzed as an instance of multidominance by several authors, including McCawley (1982), McCloskey (1986), Phillips (1996, 2003), Wilder (1999, 2008), Chung (2004), Fox & Pesetsky (2007), de Vries (2007), and Bachrach & Katzir (2009).[35] On a multidominant analysis, the shared node in (32), *questions*, has two mothers (one in each conjunct), cf. (33):[36]

[35] For arguments why a multidominant account is preferable to an analysis in terms of across-the-board rightward movement (cf. Ross 1967; Bresnan 1974; Postal 1974, 1998; Sabbagh 2007) or an ellipsis account (Wexler & Culicover 1980; Kayne 1994; Wilder 1997), I refer to the authors mentioned in the main text.

[36] For ease of exposition, I present *customers, experts* and *questions* as NPs. Given that the functional layers above N are null/empty and have no influence on the linearization statements, I ignore them here. Moreover, I make use of a ConjP (headed by the conjunction *and*) here, while Johnson (2007:18) does not (in his proposal, the two TPs are both part of a larger TP). This does not change the argumentation.

(33)



Johnson (2007) proposes, following Fox & Pesetsky (2003, 2004a) and Sabbagh (2007), that islands are linearization domains (cf. also section 3.3.1), formulating it as in (34).[37] These proposals concern strong islands (cf. subject islands in Johnson 2002 and 2009, adjunct islands in Johnson 2002 and 2009 and Fox & Pesetsky 2003, complex NPs in Johnson 2008, conjuncts in Johnson 2007, 2009, and strong WH-islands in Sabbagh 2007).

(34) *Islands as linearization domains*

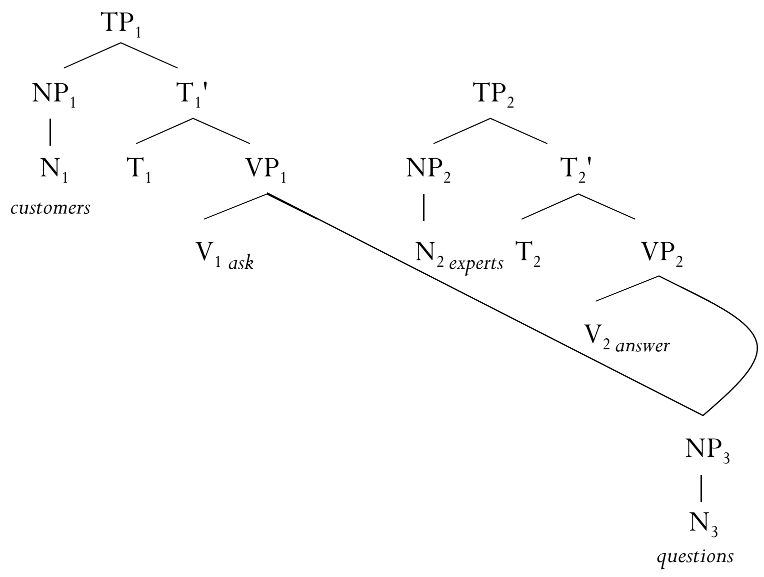
Islands are those phrases, γ , at which the linearization algorithm runs and fixes an unchangeable linearization for γ. [Johnson 2007:19]

[37] The same idea is also present in Johnson (2002, 2009), although it is given a stricter, stronger interpretation there. Order Preservation not only requires preservation of linear ordering, but also of adjacency relations. As put by Sabbagh (2007:fn.30), the idea is that, "if linear order was the only value of the syntax-phonology mapping that mattered, as with Fox & Pesetsky's theory, then string-vacuous [movement] would not be predicted to give rise to (adjunct-)island violations". As islands are not my primary concern here, I leave the correct implementation of islandhood in terms of linearization and order preservation constraints to further research.

Islands are the result of cyclically applying the algorithm that maps structures onto strings. Hence, islands must be spelled out (and cause the linearization algorithm to apply and fix a relative linear ordering among terms) before merging with their host. This is reminiscent of and extends Uriagereka's (1999) proposal to spell out complex left branches, thereby also explaining their islandhood.

According to Johnson (2007), the definition of islands in (34) will require Spell-Out and linearization at the point of the derivation of (33) where the two conjuncts have been constructed. This is because conjuncts are islands (cf. *Coordinate Structure Constraint* (CSC), Ross 1967). That point in the derivation is (35), before the two conjuncts are merged together:

(35)



The A for the TP-conjuncts in (35) is given in (36):

(36)

$$A = \left\{ \begin{array}{llll} \langle NP_1, T_1\rangle & \langle T_1, V_1\rangle & \langle NP_2, T_2\rangle & \langle T_2, V_2\rangle \\ \langle NP_1, VP_1\rangle & \langle T_1, NP_3\rangle & \langle NP_2, VP_2\rangle & \langle T_2, NP_3\rangle \\ \langle NP_1, V_1\rangle & \langle T_1, N_3\rangle & \langle NP_2, V_2\rangle & \langle T_2, N_3\rangle \\ \langle NP_1, NP_3\rangle & & \langle NP_2, NP_3\rangle & \\ \langle NP_1, N_3\rangle & \langle V_1, N_3\rangle & \langle NP_2, N_3\rangle & \langle V_2, N_3\rangle \\ & & & \\ \langle T_1', N_1\rangle & & \langle T_2', N_2\rangle & \end{array} \right\}$$

There is, however, a problem with (36), which is due to the phrase marker in (35) having two roots. Although eventually it will have only one root (cf. (33)), this is not the case at the point in the derivation depicted in (35), before the merger of the two TPs in a ConjP. Remember that Spell-Out before merger is forced by the islandhood of the two conjuncts. To quote Johnson (2007:19-20) on the issue:

> "There is no way to get a linearization out of [(36)] that meets the requirement of totality [in (9)]. Because nothing of the first conjunct asymmetrically c-commands anything in the second conjunct, the terminals in the two conjuncts are not going to get linearized relative to each other. That's because the present formulation of totality presupposed that phrase markers will have only one root node."

In a system that allows multidominant, multi-rooted phrase markers, the Totality requirement is thus in need of rephrasing. Johnson (2007:20) gives the following formulation:

(37) TOTALITY (new version)
For all distinct *x* and *y* dominated by [the same] root node in a phrase marker, either $x < y$ or $y < x$.

Recall (section 3.2) that the ordered pairs corresponding to the asymmetric c-command relations in A in (36) are disambiguated on the basis of language-particular requirements. Specifiers have to precede the material they asymmetrically c-command. Heads are linearized to the left of their complement. A maximally small subset is selected, cf. (38). The resulting linearization d(A) is given in (39). Note that, as the shared node NP_3 in (35) is dominated by two root nodes (TP_1 and TP_2), N_3 it has to be linearized twice (in each conjunct), following (37).

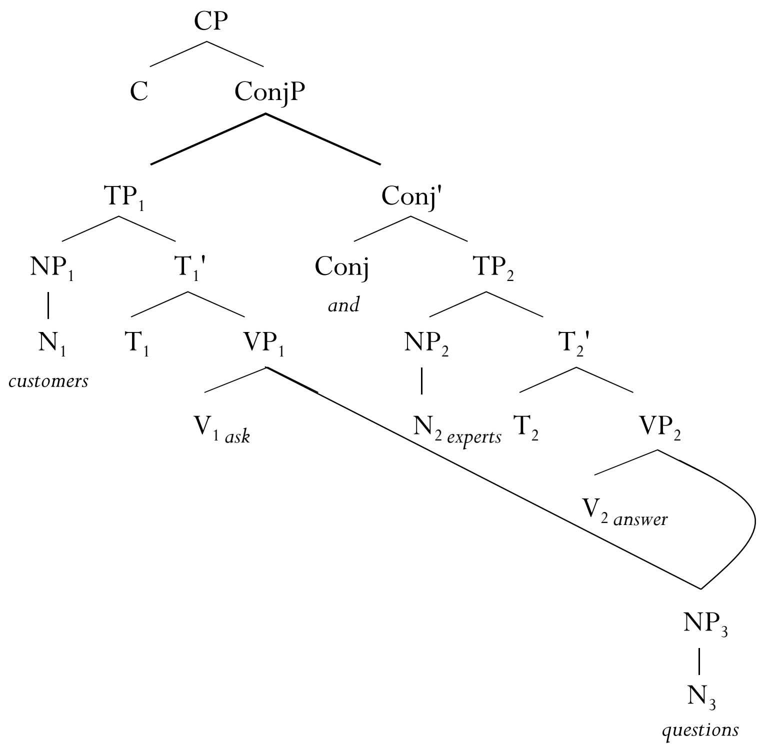
(38)
$$A' = \left\{ \begin{array}{llll} NP_1 < T_1 & T_1 < VP_1 & NP_2 < T_2 & T_2 < VP_2 \\ NP_1 < VP_1 & & NP_2 < VP_2 & \\ & & & \\ V_1 < N_3 & & V_2 < N_3 & \\ & & & \end{array} \right\}$$

(39)

$$d(A) = \left\{ \begin{array}{llll} N_1 < T_1 & T_1 < V_1 & N_2 < T_2 & T_2 < V_2 \\ N_1 < V_1 & T_1 < N_3 & N_2 < V_2 & T_2 < N_3 \\ N_1 < N_3 & & N_2 < N_3 & \\ & & & \\ V_1 < N_3 & & V_2 < N_3 & \end{array} \right\}$$

After this, the two conjuncts are merged together in a ConjP, yielding the phrase marker in (33), The C-head is merged and projects (cf. (40)), after which the root CP is spelled out. The A for (40) is given in (41).

(40)



(41)

$$A = \left\{ \begin{array}{lllll}
\langle C, TP_1\rangle & \langle C, Conj'\rangle & \langle TP_1, Conj\rangle & \langle Conj', NP_1\rangle & \langle Conj, NP_2\rangle \\
\langle C, NP_1\rangle & \langle C, Conj\rangle & \langle TP_1, TP_2\rangle & \langle Conj', N_1\rangle & \langle Conj, N_2\rangle \\
\langle C, N_1\rangle & \langle C, TP_2\rangle & \langle TP_1, NP_2\rangle & \langle Conj', T_1'\rangle & \langle Conj, T_2'\rangle \\
\langle C, T_1'\rangle & \langle C, NP_2\rangle & \langle TP_1, N_2\rangle & \langle Conj', T_1\rangle & \langle Conj, T_2\rangle \\
\langle C, T_1\rangle & \langle C, N_2\rangle & \langle TP_1, T_2'\rangle & \langle Conj', VP_1\rangle & \langle Conj, VP_2\rangle \\
\langle C, VP_1\rangle & \langle C, T_2'\rangle & \langle TP_1, T_2\rangle & \langle Conj', V_1\rangle & \langle Conj, V_2\rangle \\
\langle C, V_1\rangle & \langle C, T_2\rangle & \langle TP_1, VP_2\rangle & \langle Conj', NP_3\rangle & \langle Conj, NP_3\rangle \\
\langle C, NP_3\rangle & \langle C, VP_2\rangle & \langle TP_1, V_2\rangle & \langle Conj', N_3\rangle & \langle Conj, N_3\rangle \\
\langle C, N_3\rangle & \langle C, V_2\rangle & \langle TP_1, NP_3\rangle & & \\
\langle C, NP_3\rangle & & \langle TP_1, N_3\rangle & & \\
\langle C, N_3\rangle & & & &
\end{array} \right\}$$

The asymmetric c-command relations in A in (41) correspond to ordered pairs, the interpretation of which (in terms of precedence or subsequence) is left open. These ordered pairs need to be disambiguated, i.e. a subset has to be selected. This subset has to satisfy language-particular requirements, i.e. heads have to precede their complement and specifiers have to precede the other material in their phrase. The maximally small subset satisfying the language-specific requirements of English is given in (42). The resulting linearization d(A) is given in (43):

(42)

$$A' = \left\{ \begin{array}{lll}
C < TP_1 & TP_1 < Conj & Conj < N_2 \\
C < Conj & TP_1 < TP_2 & Conj < T_2' \\
C < TP_2 & &
\end{array} \right\}$$

(43)

$$d(A) = \left\{ \begin{array}{llll}
C < N_1 & N_1 < Conj & V_1 < Conj & Conj < N_2 \\
C < T_1 & N_1 < N_2 & V_1 < N_2 & Conj < T_2 \\
C < V_1 & N_1 < T_2 & V_1 < T_2 & Conj < V_2 \\
C < N_3 & N_1 < V_2 & V_1 < V_2 & Conj < N_3 \\
C < Conj & N_1 < N_3 & V_1 < N_3 & \\
C < N_2 & & & \\
C < T_2 & T_1 < Conj & N_3 < Conj & \\
C < V_2 & T_1 < N_2 & N_3 < N_2 & \\
C < N_3 & T_1 < T_2 & N_3 < T_2 & \\
 & T_1 < V_2 & N_3 < V_2 & \\
 & T_1 < N_3 & N_3 < N_3 &
\end{array} \right\}$$

Note that the linearization in (43) contains several problematic statements. The orderings N_3 < Conj and Conj < N_3 are antisymmetric. The statement $N_3 < N_3$ is an irreflexivity violation. Moreover, the three other orderings involving N_3 ($N_3 < N_2$, $N_3 < T_2$, $N_3 < V_2$) contradict the linearization statements that were introduced earlier in the derivation, before the two TP-conjuncts were joined in ConjP, i.e. the ones in (39). Recall that conjuncts are islands and that islands are linearization domains: the linearization established for the island cannot be changed later on (Order Preservation). Linearization statements that are introduced later in the derivation have to be both total and consistent with these earlier statements. Johnson (2007) proposes that d(A) also has the property of TOLERANCE (cf. section 3.2). That is, d(A) can contain inconsistent and conflicting linearization statements, as long as a subset of d(A) can be selected that results in a linearization that is total, antisymmetric, transitive, and irreflexive. As such, just like language-specific requirements function as what could be called filters on ordered pairs that did not get an interpretation in terms of subsequence or precedence (cf. section 3.2), the well-formedness conditions seem to function as a filter on the linearization statements in d(A).

As such, the reflexive statement can be ignored, and the three conflicting statements involving N_3 can be deleted. Moreover, one of the antisymmetric orderings can be disposed of: N_3 < Conj will be ignored, as it would otherwise result in conflicting statements and transitivity violations. For instance, the combination N_3 < Conj and Conj < N_2 would give rise to $N_3 < N_2$ (by transitivity), which is in conflict with the linearization statement $N_2 < N_3$, established earlier.[38] The remaining statements are those in (44), which will be added to the orderings established earlier (i.e. the ones in (39)).

[38] Note that, although both Johnson (2007) and Fox & Pesetsky (2003, 2004a,b) crucially adhere to the idea that later linearization statements cannot change the ones already established in previous linearization domains, there is a vital difference between the two analyses. For Fox & Pesetsky, when Spell-Out/linearization applies to a new domain, it may add new ordering statements, but it may not revise previously established ordering statements. Johnson does allow the creation of new inconsistent orderings, provided they are discarded afterwards.

(44)
$$d(A) = \left\{ \begin{array}{llll} C < N_1 & N_1 < Conj & V_1 < Conj & Conj < N_2 \\ C < T_1 & N_1 < N_2 & V_1 < N_2 & Conj < T_2 \\ C < V_1 & N_1 < T_2 & V_1 < T_2 & Conj < V_2 \\ C < N_3 & N_1 < V_2 & V_1 < V_2 & Conj < N_3 \\ C < Conj & N_1 < N_3 & V_1 < N_3 & \\ C < N_2 & & & \\ C < T_2 & T_1 < Conj & T_1 < T_2 & T_1 < N_3 \\ C < V_2 & T_1 < N_2 & T_1 < V_2 & \\ C < N_3 & & & \end{array} \right\}$$

The combination of the ordering statements in (39) and (44) results in a (correct) final linearization where the shared constituent *question* follows all material within the first conjunct and all material within the second conjunct (*Consumers ask and experts answer questions*).

Concluding, Johnson's (2007) modified linearization algorithm and, importantly, its property of TOLERANCE, allows the creation of inconsistent ordering statements provided there is a subset of those orderings that results in a linearization that is total, antisymmetric, transitive, and irreflexive. Moreover, Order Preservation requires that linearization statements that are introduced later in the derivation be both total and consistent with earlier statements. This allows multidominant phrase markers and grammatical linearization statements in a cyclic Spell-Out/linearization model of the grammar.

4 Ellipsis

The term *ellipsis* covers a range of phenomena where, under certain conditions, a part of a clause 'goes missing' (is left unpronounced). The sentence in (45) is an example of *VP-ellipsis*, a process that elides verb phrases. The sentence in (46) illustrates *sluicing*, where ellipsis targets clauses (TPs). Ellipsis is marked by ⟨—⟩.

(45) VP-ellipsis
a. I prefer to go paperless and apparently Apple does too.[39]
b. … Apple does ⟨$_{VP}$ ~~prefer to go paperless~~ ⟩ too.

[39] http://online.worldmag.com/2011/08/05/world-magazine-for-ipad-subscriptions-now-available/

(46) Sluicing
a. For those that can't see it, Adam just singlehandedly blew the peloton to pieces and all of a sudden Astana and CdE started hammering along. Someone must be missing, but I don't know who.[40]
b. … but I don't know who ⟨$_{TP}$ ~~is missing~~ ⟩.

In spite of the fact that linguistic material is absent in an elliptical sentence, the meaning of the sentence is perfectly clear. The sluicing example in (46), for instance, is interpreted as containing the full embedded constituent question *but I don't know who is missing*, even though an interrogative clause is not phonetically realized; the only element overtly following the verb *know* is the WH-word *who*. Thus, the interpretation of an elliptical sentence is richer than what is actually pronounced. That is, although ellipsis sites are phonetically empty, they are interpreted at LF.

4.1 Three approaches to ellipsis[41]

The question of how meaning can arise in the absence of form has been answered in different ways in the literature. A first group of proposals (cf. van Riemsdijk 1978; Ginzburg & Sag 2000; Culicover & Jackendoff 2005) adheres to a non-structural approach to ellipsis: unpronounced material is taken to be syntactically absent. It is assumed that the syntax matches the phonology: the structure simply ends with the last element before the ellipsis site. For the sluicing example in (46), the CP-node of the complement of the verb *know* directly dominates (only) the WH-DP *who*, as represented in (47)a. Approaches like these turn to semantic and pragmatic mechanisms to supply the desired meaning.

Structural approaches, on the other hand, take an ellipsis site to contain (unpronounced) syntactic structure. Only phonological information is missing. These proposals can be divided into two classes. A first 'structural' group of proposals posits that the gap in an elliptical sentence is an empty, structureless category (a null proform) that is interpreted at LF (Hardt 1993; Fiengo & May 1994; Lobeck 1995; Chung et al. 1995; Wilder 1997; Depiante 2000). For (46), a null proform of the category TP is merged as part of a CP, the specifier of which is

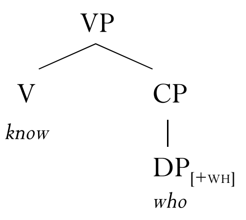
[40] http://fairwheelbikes.com/forum/viewtopic.php?f=127&t=6980&start=15

[41] This subsection is based largely on overviews in Bartos (2001), Merchant (2001, 2005), Winkler (2006), Aelbrecht (2009), and van Craenenbroeck (2010).

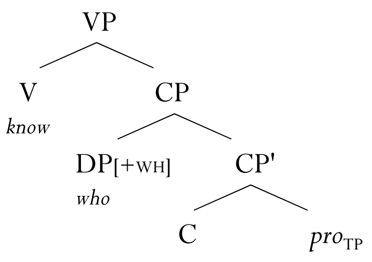
occupied by the base-generated WH-phrase, cf. (47)b. At LF, this null proform is linked to its antecedent. Either the null proform is taken to get its meaning from the antecedent by general mechanisms governing the recovery of meanings from context (the same mechanisms by which pronouns get their meaning from an antecedent), or the LF of the antecedent is posited to be copied into the empty category at LF.

A second 'structural' group takes the elided material to be completely present at the syntactic level: ellipsis targets a full syntactic structure. A fully-fledged sentence is generated in the computational system and handed over to the interfaces, PF and LF. At PF, the phonological component is instructed to delete part of it (i.e. leave part of it unpronounced) (Ross 1969; Sag 1976; Lasnik 1999, 2001; Johnson 2001; Merchant 2001 *et seq.*; Aelbrecht 2009; van Craenenbroeck 2010).[42] These proposals take the verb *know* in (46) to select a fully-fledged complement CP, the TP-complement of which is deleted at PF, cf. (47)c. As noted by van Craenenbroeck (2010:1), under the PF-deletion account, "the interpretation of an elliptical sentence proceeds exactly as that of a nonelliptical one, that is, via a compositional, one-to-one mapping between syntax and semantics."

(47) a. NON-STRUCTURAL

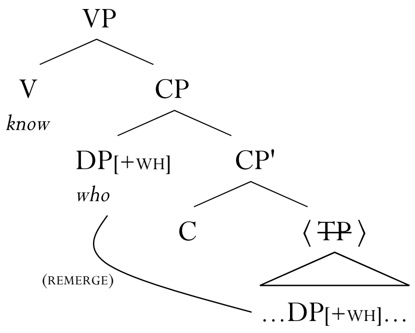


b. STRUCTURAL: NULL PROFORM



[42] A similar approach to ellipsis is the view that ellipsis is radical deaccenting. Phonological deletion is then considered an alternative option to mere deaccenting (cf. Tancredi 1992; Chomsky & Lasnik 1993; Sauerland 1996). For differences between deaccenting and deletion, see e.g. Merchant (2001).

c. STRUCTURAL: PF-DELETION



I adopt the third approach to ellipsis, i.e. a structural approach that takes ellipsis to be a PF-process. In the phonological component, material that is present in the syntactic output is deleted.

In spite of their name, PF-deletion accounts do not necessarily involve actual deletion. That is, they do not always postulate actual deletion rules for the phonological content of an ellipsis site. For instance, under a Late Insertion model of the grammar (cf. Distributed Morphology (DM), Halle & Marantz 1993; Harley & Noyer 1999; Embick & Noyer 20081, 2007), there are no phonological features in syntax. Syntactic terminals are purely abstract, consisting of syntactic/semantic features and feature bundles, and having no phonological shape. Only after syntax, in the mapping to PF, the phonological expression of syntactic terminals is provided through the operation of *Vocabulary Insertion* (cf. Harley & Noyer 1999:3). At Vocabulary Insertion, a list of lexical items (the Vocabulary) is accessed. A Vocabulary item is a relation between a phonological signal and information about where that piece is to be inserted, i.e. about which abstract morphemes it can express. At Vocabulary Insertion, a phonological shape is matched to an insertion point (i.e. a feature (bundle) in a terminal node). Under this framework, it has been proposed that 'deletion' is lack of (late) lexical insertion (cf. Wilder 1997; Bartos 2000, 2001; Kornfeld & Saab 2004; Aelbrecht 2009; Saab 2009; van Craenenbroeck 2010:253,fn.2; Schoorlemmer & Temmerman 2010). That is, Vocabulary Insertion into the terminal nodes contained within an ellipsis site is blocked.[43] Terminal

[43] The proposal that ellipsis blocks lexical insertion rules is also a consequence of (i):

(i) *The Ellipsis-Morphology Generalization* [cf. Saab & Zdrojewski 2010; Lipták & Saab 2012]
For every morphological operation MO that affects the domain of X, where X contains the target of MO, MO cannot apply in X if X is subject to ellipsis.

feature bundles in an ellipsis site are not matched to phonologically interpretable content. In a non-insertion approach, a terminal does not receive phonological contents in the first place; in a deletion approach, a terminal does get phonological contents, but loses it at a subsequent point in the PF-branch. Bartos (2001:8-9) notes that distinguishing between phonological non-insertion and phonological deletion is "far from obvious, i.e. one cannot make any theory-external or empirical difference between the two". The choice of mechanism is dictated by theoretical assumptions (on the relation between lexicon and derivation, on the operations available in the PF-branch, etc.). In this dissertation, I assume a strictly non-lexicalist (DM) model of the syntax-morphology interface and therefore, I take ellipsis to be non-insertion.[44,45]

Specifically, I follow Fox & Pesetsky's (2003, 2004a) interpretation of ellipsis/PF-deletion, given in (48).

(48) *Ellipsis*

Ellipsis of α involves (i) the non-pronunciation of any terminal element dominated by α and (ii) the deletion from the Ordering Table of all ordering statements referring to the terminal elements dominated by α.

[cf. Fox & Pesetsky 2003:21]

The notion of 'Ordering Table' in (48) is defined by Fox & Pesetsky (2003:16) as follows: "An *Ordering Table* receives the output of [the linearization algorithm] at various points as the derivation proceeds. The information that the Ordering Table receives from [Linearization] at any given stage is added to the information already present in the "Ordering Table." What sets this definition apart from other 'deletion/non-insertion' proposals is that ellipsis not only affects the pronunciation

[44] Nevertheless, the analysis in this dissertation can be made compatible with an actual deletion-view of ellipsis as well.

[45] According to Andrés Saab (p.c.), there *is* evidence that deletion analyses and non-insertion approaches are empirically distinguishable. The empirical evidence for instance comes from the (in)sensitivity of phonological and (late) morphological features to the identity condition on ellipsis (thus extending and generalizing the more traditional conception of the identity condition on ellipsis that conceives identity as an LF or post-LF condition). This is because under the deletion approach, but not under the non-pronunciation one, there is a point in the derivation in which such features are present (i.e. before deletion). The evidence suggests that phonological and late morphological features are fully insensitive for the identity condition on ellipsis, which favors a non-pronunciation approach over one in terms of deletion. Saab (p.c.) stresses – and I agree – that the right way to proceed, methodologically speaking, is comparing both approaches under every possible formulation of the identity condition (as in Saab 2009). If it turns out that the non-insertion approach is preferable to the deletion approach, this supports the point of view adopted in this dissertation.

of terminal elements, it also targets the linearization statements mentioning these terminals.

Fox & Pesetsky (2003, 2004a) seem to imply that when a node is targeted by ellipsis, ordering statements referring to the terminals dominated by this node are first generated, but deleted later on by ellipsis. In what follows, I will give (ii) in (48) a slightly different interpretation, though. Suppose ellipsis targets a node α. Then, the ordering statements referring to the terminals dominated by α that were generated earlier in the derivation are deleted. New ordering statements are, however, not generated and added to the Ordering Table (only to be deleted just afterwards). Landau (2004) and Kandybowicz (2007) argue that economy principles disfavor pronouncing elements that are unnecessary at the PF interface level. This reasoning can be straightforwardly extended to ordering statements that are unnecessary at the PF interface.

4.2 The timing of ellipsis

4.2.1 BACKGROUND: LICENSING OF ELLIPSIS[46]

Linguistic material cannot always be omitted, even when the interpretation is clear in the context. Examples are given in (49): even though the noun *dress* in (49)a and the verb *arrived* in (49)b are recoverable from the context, they cannot be elided.

(49) [Aelbrecht 2009:15]
a. I bought the red dress and Alice bought the blue *(dress).
b. * Max having arrived and Morgan not having, we decided to wait.

Likewise, the distribution of a particular elliptical phenomenon across languages can be quite limited. Lobeck (1995) and Aelbrecht (2009) note, for instance, that while the auxiliary *have* in English allows its VP-complement to be elided, its equivalent in a language like Italian does not.

(50) [Aelbrecht 2009:15]
a. Monica has paid already, but Alice hasn't.
b. * Antonio ha già pagato, ma Stefano non ha ancora.

b. *	Antonio	ha	già	pagato,	ma	Stefano	non	ha	ancora.
	Antonio	has	already	paid	but	Stefano	not	has	yet [Italian]

[46] The discussion in this subsection is based mainly on Johnson (2001) and Aelbrecht (2009).

These examples show that ellipsis can only occur in specific syntactic environments. This is referred to as the *licensing condition* on ellipsis. The specific licensing criteria are dependent on the elliptical phenomena under consideration, and on language particular properties. In English (50)a, a finite auxiliary (*has*) syntactically licenses VP-ellipsis. The Italian finite auxiliary *ha* in (50)b, on the other hand, does not license VP-ellipsis.

Following Zagona (1982, 1988a, 1988b), Lobeck (1992, 1995), Martin (1992, 1996), Johnson (2001), Merchant (2001 *et seq*.), and Aelbrecht (2009), amongst others, I take ellipsis to require a licensing head. That is, only particular heads with a certain feature specification, henceforth licensors, can trigger PF-deletion: their complements constitute the ellipsis site.[47] For VP-ellipsis in English, for instance, the licensing head is generally taken to be the inflectional head T (when it is occupied by a finite auxiliary, a modal, or the infinitival marker *to*), see the discussion in section 4 of chapter 3 (see also sections 2.3 and 3.2 of chapter 4).[48]

4.2.2 DERIVATIONAL ELLIPSIS (AELBRECHT 2009)

As discussed in the previous subsection, ellipsis is licensed by a licensor, a head carrying a certain morpho-syntactic feature specification. The question arises at which point in the derivation this licensor's complement is actually elided. As noted by Aelbrecht (2009:107), there are two possible implementations. Either the effect of merging the licensing head, i.e. ellipsis of its complement, is postponed until the derivation is finished, or ellipsis occurs immediately, as soon as the licensor is merged. Aelbrecht (2009:Ch.3.2) argues in favor of the latter option, i.e. ellipsis is derivational:

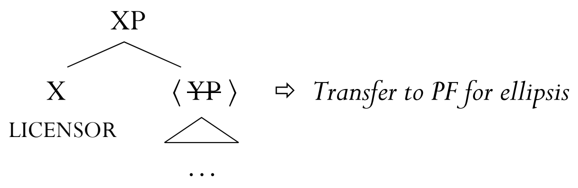
(51) Ellipsis occurs in the course of the derivation, as soon as the licensing head is merged. At this point the ellipsis site becomes inaccessible for any further syntactic operations, and vocabulary insertion at PF is blocked.
[Aelbrecht 2009:91, (1b)]

[47] I will not go into the various proposed implementations of the fact that ellipsis needs a licensing head. What matters here is solely that there is a licensing head, which triggers PF-deletion of its complement. See Lobeck (1992, 1995), Merchant (2001 *et seq*.), and Aelbrecht (2009), amongst others, for proposals on how to implement the licensing condition.

[48] See Zagona (1982, 1988a, 1988b), Lobeck (1992, 1995), Martin (1992, 1996), Johnson (2001), Aelbrecht (2009).

Thus, the effect of ellipsis is twofold. First, it marks the ellipsis site – i.e. everything c-commanded by the licensor – for ellipsis at PF.[49] As soon as the licensing head (X in (52)) is merged, its complement (YP in (52)) is immediately sent off to PF, marked for ellipsis. As a result, PF refrains from pronouncing anything inside the ellipsis site. For Aelbrecht (2009), this implies that lexical insertion into the terminal nodes of this part of the structure is blocked.

(52)



Second, at the moment a constituent is marked for ellipsis by a licensing head, the ellipsis site is frozen for any further syntactic operations.[50] This is because the ellipsis site has been sent off to PF, where it is inaccessible to narrow syntax. Aelbrecht (2009:Ch.3) presents supporting evidence concerning extraction data in Dutch modal complement ellipsis: extraction out of the ellipsis site is only possible up until the moment when the licensor is merged.[51] As syntactic freezing of the ellipsis site is not the primary concern here, I refer the reader to Aelbrecht (2009) for details and relevant data.

In this dissertation, I adopt Aelbrecht's (2009) proposal that ellipsis takes place in the course of the derivation.[52] The ellipsis site is sent off to PF as soon as the licensing head is merged. Adopting Fox & Pesetsky's (2003, 2004a) view on ellipsis, given in (48) and repeated here, this means that the terminals in the complement of the licensor are left unpronounced and all linearization statements mentioning them are immediately deleted at PF.

[49] The details in Aelbrecht (2009:Ch.3.2) are somewhat different. According to Aelbrecht, it is not always the complement of the licensor that is elided. She implements ellipsis licensing via Agree, which is not taken into consideration here. I gloss over these details here. See section 2.2 of chapter 4 for some related discussion, though.

[50] Here, Aelbrecht (2009) deviates from 'traditional' PF-deletion accounts.

[51] Aelbrecht (2009) argues that the licensor itself can attract an element out of the ellipsis site prior to ellipsis, as all operations triggered by the same head take place simultaneously.

[52] To be precise, and as should be clear from the foregoing discussion, it is the *licensing/marking/shipping off of* ellipsis that takes place in the course of the derivation. The reader should keep in mind that this is what I mean when I talk about "ellipsis in the course of the derivation" in the following chapters.

(48) *Ellipsis*
Ellipsis of α involves (i) the non-pronunciation of any terminal element dominated by α and (ii) the deletion from the Ordering Table of all ordering statements referring to the terminal elements dominated by α.
[cf. Fox & Pesetsky 2003:21]

This is perfectly in line with the cyclic view of the grammar adopted here. Just like phasal domains and complex left branches (cf. section 3.3.1), ellipsis sites are shipped off to PF in the course of the derivation. There is one crucial difference, however. While phasal domains and complex left branches are linearized during the derivation (i.e. they add ordering statements to the Ordering Table), transfer of an ellipsis site results in the deletion of linearization statements (and non-pronunciation).