⋆ BRIEFS ⋆

Recursion as Derivational Layering: An Amendment to Zwart

Andreas Trotzke & Antje Lahne

1. Introduction

In a recent issue of *Biolinguistics*, Zwart (2011a) argues against the prominent view that recursion in natural language should be understood in terms of syntactic embedding and proposes that recursion is instead evidenced by derivational layering: The more derivational cycles (= layers) are needed to derive a structure S, the more recursive S is. Derivational layering is also taken as a measure for complexity: The more derivational layers are needed to derive a structure S, the more complex S is. According to Zwart, this model correctly predicts that recursive center embedding of a certain type (which we will call type A) is more recursive and thus more complex than two other types of center embedding (types B and C) as its computation involves more derivational layers.

In this amendment, we will first point out how the notion of recursion within the derivational layering approach differs from other prominent concepts of recursion. In Section 3, we will demonstrate that the approach does not make the prediction assumed by Zwart; rather, the model predicts that type B center embedding structures involve an equally high number of derivational layers as type A structures. Section 4 concludes and points out the consequences of this amendment for theories based on derivational layering as an indicator for recursion.

2. Recursion as Derivational Layering

In current minimalist theory, recursion is encapsulated in the basic operation Merge. As (1) illustrates, Merge recursively generates syntactic objects, since it takes the output of a previous application (1a) as part of its input when running the next time (1b):

We thank the audiences at UQAM, University of Potsdam, and University of Stuttgart, as well as Markus Bader, Josef Bayer, Jan-Wouter Zwart, and an anonymous reviewer for useful comments and discussion. We gratefully acknowledge financial support from the German Academic Exchange Service (DAAD) and from the German Excellence Initiative (Young Scholar Fund, University of Konstanz, project no. 65411).

(1) a.
$$\begin{tikzcd} & \{\,\alpha\,\{\,\alpha,\beta\,\}\,\} & \\ \alpha \arrow[r, leftarrow] & {} \arrow[u] \arrow[r] & {}_{\text{Merge}}\ \beta \end{tikzcd}$$

b.
$$\begin{tikzcd} & \{\,\alpha\,\{\,\gamma\,\{\,\alpha\,\{\,\alpha,\beta\,\}\,\}\,\}\,\} & \\ \{\,\alpha\,\{\,\alpha,\beta\,\}\,\} \arrow[r, leftarrow] & {} \arrow[u] \arrow[r] & {}_{\text{Merge}}\ \gamma \end{tikzcd}$$

Due to the category-independent, derivational nature of this approach to structure-building, understanding recursion in terms of Merge renders most of the recent debates on the universality of syntactic recursion a non-issue because this notion of recursion implies that "a language that lacks recursion would be considerably [...] exotic. No sentence in such a language could contain more than two words" (Nevins *et al.* 2009: 366, n. 11).

Another prominent notion of recursion does not make use of the relational understanding of projection levels. This concept has a strong representational aspect in that it refers to specific, categorially defined types of syntactic embedding (cf. Everett 2005 and subsequent work). For instance, the right-branching structure in (2) is said to imply recursion, since a CP is embedded in another CP:

(2) $[_{CP}$ John thinks $[_{CP}$ that Mary said $[_{CP}$ that it was raining $]\,]\,]$

Zwart (2011a) diverges from both notions of recursion. As opposed to the 'recursion-as-Merge' view, he proposes the operation Split-Merge, which works iteratively rather than recursively. That is, according to Zwart's approach, the basic generative procedure can be carried out by a finite-state machine because the derived sequences are computed in a strictly local manner: by simple iteration (cf. Abelson & Sussman 1984: 29–33). More precisely, Split-Merge works from top down and from left to right, iteratively splitting off identified elements from the residue set, that is, the Numeration N (cf. Zwart 2009):

(3) Split-Merge

a. $N = \{\,\alpha, \beta, \gamma\,\}$

b. Merge: split $x \in N$ off from N, yielding $\langle\, x, N - x \,\rangle$

c. Merge α yielding $\langle\, \alpha\,\{\,\beta, \gamma\,\}\,\rangle$

d. Merge β yielding $\langle\, \alpha\,\langle\, \beta\,\{\,\gamma\,\}\,\rangle\,\rangle$

e. Merge γ yielding $\langle\, \alpha\,\langle\, \beta\,\langle\, \gamma\,\{\quad\}\,\rangle\,\rangle\,\rangle$

At each step in the derivation, Split-Merge turns the numeration into an ordered pair $\langle\, x, y\,\rangle$, where $x \in N$ and $y = (N - x)$. A general condition on the derivation is that at each derivational step, the residue { _ } must form a constituent, in the sense of meeting any of the classical constituency tests (substitution, ellipsis, movement/topicalization, clefting, and coordination).[1] It follows that, as soon as a sequence involves a complex non-complement (as in [*The dog*] *barks*), this sequence cannot be derived by splitting off the leftmost element of the complex non-complement, since the residue ({*dog barks*}) does not form a constituent.

[1] Of course, this condition must be seen in the context of more fundamental assumptions of Zwart's top-down approach (such as the relation between N and the workspace). Since a full explication of the model would take us too far afield, we refer the reader to the programmatic paper Zwart (2009).

Therefore, complex non-complements must be derived in a separate derivational cycle, that is, a subderivation of [*the dog*], which is then integrated into the Numeration as an atom (cf. Uriagereka 1999).[2] In the model described here, each derivational cycle is called *derivational layer*.[3]

In opposition to the 'recursion-as-syntactic-embedding' view Zwart (2011a) claims that recursion is not necessarily evidenced by embedding; rather, he argues that recursion in language should be understood in terms of derivational layering.[4] According to Zwart, derivations are layered because the output of a previous derivation can function as one single item in the numeration of the next derivation. This derivational procedure is recursive in the sense that its output is part of the input of the same procedure. Crucially, if a sequence is derived in one derivational layer (as it is in example (2)), it does not imply recursion. In contrast, simple clauses containing complex specifiers (like *The dog barks*) imply recursion.

Assuming this notion of recursion, Zwart argues throughout his paper "that we cannot tell that a grammar is recursive by simply looking at its output; we have to know about the generative procedure" (Zwart 2011a: 43). In what follows, we demonstrate that in one case, the paper actually does draw conclusions that refer to the output (representation) rather than to the procedure (derivation). In other words, within the derivational layering model, recursion is understood in terms of embedding again, not in terms of derivational layering. The consequence of this change of perspective is that the derivational layering model is attributed a prediction about the complexity of center-embedded clauses. We show, by strictly adhering to the derivational procedure, that the model does actually not make this prediction.

3. Amendment

Zwart (2011a) makes a statement about the grades of complexity of different kinds of center embedding. The examples in (5) exemplify the core data:

(5)	a.	The dog **the cat** *the man kicked* **bit** barked.		(A)
	b.	The dog t**hat the cat bit** *that the man kicked* barked.	(25)	(B)
	c.	The dog **that bit the man** *that kicked the cat* barked.	(26)	(C)

The examples in (5) differ in that (5a) involves self-embedding, whereas (5b) and

2 The mechanism described here yields the effect of the Extension Condition (cf. Chomsky 1995) while actually stating it from a top-down perspective: According to the Extension Condition, "Merge always applies at the simplest possible form: at the root" (Chomsky 1995: 248). Accordingly, *dog* and *barks* cannot be merged, excluding *the*, as *the* would then have to be merged with the non-root node *dog*. In the Split-Merge model, *the* cannot be split off from the residue {*dog barks*}, as the residue does not form a constituent.

3 Interestingly, the derivational layering approach has a feature that moves it close to so-called 'dynamic' approaches to phases (as proposed in Svenonius 2001, den Dikken 2007, Gallego & Uriagereka 2007, Gallego 2010): Derivational layers are not defined in terms of syntactic categories; for example, a CP may be the output of a separate derivation layer, but it does not have to be (cf. (2)). Thus, phase status is not a fixed property of certain heads (such as C).

4 The layers themselves, as already mentioned above, are generated by the operation Split-Merge, which works iteratively rather than recursively.

(5c) contain right-branching.[5] Given these output-related observations, Zwart claims that type A center embedding is more complex (i.e. contains more derivational layers) than type B and type C, which are equal in complexity. In particular, he states that "the difference may be accounted for by the fact that (25) and (26) contain fewer derivational layers (due to right-branch embedding) and hence less recursion" (Zwart 2011a: 49). However, when running through the computation, we noticed that type B does not contain fewer derivational layers than type A; (6) shows the results of applying Split-Merge to all three types (see the appendix for the detailed derivations):[6]

(6) a. [[[The dog] [[**the cat**] [*the man*] *kicked*] **bit**] barked] A: 6 layers
b. [[[The dog] **that** [[**the cat**] **bit**] *that* [*the man*] *kicked*] barked] B: 6 layers
c. [[[The dog] **that bit** [**the man**] *that kicked the cat*] barked] C: 4 layers

As (6) illustrates, type A center embedding patterns with type B in that they involve an equal number of derivational layers; they are thus equally recursive and equally complex, as opposed to type C, which is less complex. Of course, where type B has complex subjects ([*the cat*], [*the man*]), type C has gaps (as the subjects are relativized: *that Ø bit the man*, *that Ø kicked the cat*). But the distinction between B and C concerning their derivational complexity cannot be removed by replacing *the cat* etc. by, for instance, *cats*, as suggested by Jan-Wouter Zwart (p.c.), because then we arrive at:

(7) a. [[[The dog] [cats people kick] bite] barked] A: 4 layers
b. [[[The dog] that [cats bite] that people kick] barked] B: 4 layers
c. [[[The dog] that bites people that kick cats] barked] C: 3 layers

What is instead relevant here is the head-finality of type B. So, both [*the cat bit*] and [*cats bite*] constitute separate subderivations, since they imply the following derivational steps, where *bit that the man kicked* and *bite that people kick* do not form constituents:

(8) a. * 〈 [the dog] 〈 that 〈 [the cat] { bit, that, the, man, kicked } 〉 〉 〉 (6b)
b. * 〈 [the dog] 〈 that 〈 cats { bite, that, people, kick } 〉 〉 〉 (7b)

In English, of course, this head-finality is the result of fronting the object. Unfortunately, it is not clear so far how to implement movement phenomena in the Split-Merge approach. Zwart (2009: 181) concedes that in this respect the derivational layering system is very tentative and that "[o]ther approaches, mimicking movement, might also be pursued". Accordingly, although we have shown that Zwart's claim that type A is more complex than type B cannot be upheld without

[5] To facilitate reading, we call the center embedded configuration that contains self-embedding *type A* center embedding (5a), and those that do not contain self-embedded relative clauses *type B* (5b) and *type C* (5c) center embeddings, respectively.

[6] Note that the bracketing does not indicate constituency but derivational layers. We follow Zwart in using this notation.

further assumptions about movement, the fact that the Split-Merge model is very preliminary in this regard makes the claim a moving target for us. However, when turning to German, we will show now that Zwart is forced to assume an additional layer in type B structures anyway, and thus concurs with the results we find for type A and type B structures in this language:

(9) *German*

a. [[Fido [der Hans der Eva liebt] beißt] bellt] A: 3 layers
Fido who Hans who Eva loves bites barks
'Fido, who bites Hans, who loves Eva, barks.'

b. [[Fido der [Hans beißt] der Eva liebt] bellt] B: 3 layers
Fido who Hans bites who Eva loves barks
'Fido, who bites Hans, who loves Eva, barks.'

Zwart (2011b) assumes, in contrast to Kayne (1994), that not all head-finality is 'pseudo-finality' and that, therefore, languages like Dutch and German show head-final linear order in relative clauses without any movement. Crucially now, Zwart claims that this head-final order implies a separate derivation, but not because of structural (configurational) reasons; rather, he proposes a separate derivation because of 'interpretive criteria'. Note that Zwart (2009: 173) assumes that there are two criteria for determining that an element is the output of a subderivation: (i) configurational criteria (the condition of yielding constituents, as sketched above in section 2) and (ii) interpretive criteria (showing interface effects). Zwart (2011b) demonstrates, based on many syntactic observations from Dutch, that head-final linear order (in Dutch) shows interface effects (idiosyncratic sound/meaning pairings), and that, therefore, head-finality implies an additional derivational layer. Many of the Dutch facts also hold for German. For example, nonspecific indefinites like *ein* lose their nonspecific reading as soon as they are not adjacent to the verb:

(10) *German*

a. … dass Hans selten ein Buch liest
that Hans rarely a book read
'that Hans rarely reads a book.'

b. … dass Hans ein Buch selten liest
that Hans a book rarely read
'… that Hans rarely reads a book.'
(= 'What Hans rarely does to a book is read it.')

Based on such facts, Zwart (2011b: 115) proposes the generalization that "[h]ead-finality is a linguistic sign, signaling derivation layering".[7] If this generalization

[7] In accordance with a reviewer, we would like to point out that this looks more like the beginning of a theory and not like a generalization. In particular, Zwart states that head-finality is a sign for the presence of an additional layer without formally implementing how exactly the layer is actually triggered.

holds, then type B center-embedding (at least in languages like German) involves an additional derivational layer anyway, due to 'interpretive criteria', so that type A and type B involve an equal number of derivational layers and are thus equally recursive. However, we have demonstrated that type B implies the same amount of subderivations because of *configurational criteria*, namely because of the non-local dependency between antecedent and relative pronoun (i.e., when the antecedent [*the cat*] is split off, the residue *bit that the man kicked* does not form a constituent).

In the last section of this amendment, we will point out consequences of this finding for theories based on derivational layering as an indicator for recursion.

4. Conclusion and Outlook

In this amendment, we first have shown how the derivational layering model differs from two other prominent notions of recursion. After that, we have demonstrated that the derivational layering approach does not make the prediction assumed by Zwart; rather, the model predicts that type B center embedding structures involve an equally high number of derivational layers as type A structures. To make this finding immune to rejections that could be raised due to the somewhat vague status of syntactic movement in Zwart's theory, we have shown that the same ratio of derivational layers between type A and type B structures holds for German. We ended with the observation that Zwart assumes that type B structures imply the same amount of layers as type A structures due to interpretive criteria, while we have shown that this is the case due to purely configurational criteria. As a last point, let us briefly sketch why it might be important to note that type A and type B structures are equally recursive because of configurational criteria.

First of all, Zwart (2011a) suggests drawing inferences between the amount of derivational layers and processing constraints of center-embedded structures. According to him, "center-embedding cannot be performed indefinitely, unlike right-branch embedding […]. It seems, then, that recursion (as understood here) comes with a cost" (Zwart 2011a: 49). Of course, this difference in 'cost' (memory load) refers to the configurational situation that type A involves two non-local dependencies between subjects and the respective verbs, while type B and C only involve one dependency of this type. In general, we find nothing objectionable about drawing those inferences, especially since Split-Merge works top-down and from left to right, and it is well known that this derivational perspective can contribute to a theory of the interface between grammar and the parser (cf. Phillips 1996, 2003, Weinberg 1999). However, coming back to our finding again that type B is equally recursive as type A, Split-Merge gives significance to the non-local dependency between antecedents and relative pronoun and thereby reflects the psycholinguistic insight that this relation plays a significant role in affecting relative clause extraposition in German (cf. Shannon 1992, Uszkoreit *et al.* 1998). So, while the general plausibility of Split-Merge concerning performance preferences can be maintained, we arrive at a more fine-grained picture than Zwart.

With this issue in mind, we would like to conclude by suggesting a distinction between a notion of recursion that is related to basic ('atomic') structure building and a more global notion of recursion that refers to the whole (sub-) derivation. In particular, there is good reason to assume that the grammar is recursive in a more elementary sense because "[t]he fact that Merge iterates without limit is a property at least of LIs […]. EF [edge feature] articulates the fact that Merge is unbounded, that language is a recursive infinite system of a particular kind" (Chomsky 2008: 139). We propose that this notion coexists with the concept of recursion exemplified by Zwart's approach of derivational layering. This notion of recursion is compatible with other recent approaches within minimalism: A widely assumed view is that Spell-Out is a cyclic procedure targeting the complements of (certain) heads. Thus, after Spell-Out, the derivation is split up into chunks reduced to their head and their left edge. These structural primitives are inserted in the numeration for the next derivational step. This property of the grammar has already led other scholars to claim, although for different reasons than Zwart, that recursivity arises from the cyclicity of derivations, or, in other words, that recursion is organized 'phasally' (cf. especially Arsenijević & Hinzen 2010, Hinzen & Arsenijević to appear). Given that recursion, in these approaches, is dealt with as an interface phenomenon, it is reasonable to assume that this form of recursion is (at least partly) subject to principles that belong to external components like the performance systems (cf. Trotzke *et al.* submitted). In this regard, it is not surprising at all that type B structures are more complex than type C structures and (at least regarding the ratio of derivational layers) equally complex as type A structures, given the psycholinguistic insight mentioned above and given that, at least in German, the difference between type A and type B structures concerning performance preferences seems to be less significant than widely assumed (cf. Bader 2011).

In sum, understanding recursion as derivational layering may point toward a notion of recursion that can be informed by processing data, and thus this perspective offers a promising interdisciplinary domain of research within the biolinguistic approach to language.

Appendix

This appendix provides the detailed derivations of all three center-embedding types given in (6); for further explication, please contact the authors.

A. Type A: 6 layers

(6) a. [[[The dog] [[**the cat**] [*the man*] *kicked*] **bit**] barked]

Derivation:

(A1) a. N = {the, dog, the, cat, the, man, kicked, bit, barked}
b. *⟨the {dog, the, cat, the, man, kicked, bit, barked}⟩

(A2) a. N = {the, dog}
b. ⟨the {dog}⟩
c. ⟨the ⟨dog { }⟩⟩ => derivational layer

(A3) a. N = {[the dog], the, cat, the, man, kicked, bit, barked}
b. * ⟨[the dog] {the, cat, the, man, kicked, bit, barked}⟩

(A4) a. N = {[the dog], the, cat, the, man, kicked, bit}
b. ⟨[the dog] {the, cat, the, man, kicked, bit}⟩
c. * ⟨[the dog] ⟨the {cat, the, man, kicked, bit}⟩⟩

(A5) a. N = {the, cat}
b. ⟨the {cat}⟩
c. ⟨the ⟨cat { }⟩⟩ => derivational layer

(A6) a. N = {[the dog], [the cat], the, man, kicked, bit}
b. ⟨[the dog] {[the cat], the, man, kicked, bit}⟩
c. * ⟨[the dog] ⟨[the cat] {the, man, kicked, bit}⟩⟩

(A7) a. N = {[the cat], the, man, kicked}
b. ⟨[the cat] {the, man, kicked}⟩
c. * ⟨[the cat] ⟨the {man, kicked}⟩⟩

(A8) a. N = {the, man}
b. ⟨the {man}⟩
c. ⟨the ⟨man { }⟩⟩ => derivational layer

(A9) a. N = {[the cat], [the man], kicked}
b. ⟨[the cat] {[the man], kicked}⟩
c. ⟨[the cat] ⟨[the man] {kicked}⟩⟩
d. ⟨[the cat] ⟨[the man] ⟨kicked { }⟩⟩⟩ => derivational layer

(A10) a. N = {[the dog], [[the cat] [the man] kicked], bit}
b. ⟨[the dog] {[[the cat] [the man] kicked], bit}⟩
c. ⟨[the dog] ⟨[[the cat] [the man] kicked] {bit}⟩⟩
d. ⟨[the dog] ⟨[[the cat] [the man] kicked] ⟨bit { }⟩⟩⟩
=> derivational layer

(A11) a. N = {[[the dog] [[the cat] [the man] kicked] bit], barked}
b. ⟨[[the dog] [[the cat] [the man] kicked] bit] {barked}⟩
c. ⟨[[the dog] [[the cat] [the man] kicked] bit] ⟨barked { }⟩⟩
=> derivational layer

=> 6 derivational layers

B. Type B: 6 layers

(6) b. [[[The dog] **that** [[**the cat**] **bit**] *that* [*the man*] *kicked*] barked]

Derivation:

(B1) a. N = {the, dog, that, the, cat, bit, that, the, man, kicked, barked}
b. * ⟨the {dog, that, the, cat, bit, that, the, man, kicked, barked}⟩

(B2) a. N = {the, dog}
b. ⟨the {dog}⟩
c. ⟨the ⟨dog { }⟩⟩ => derivational layer

(B3) a. N = {[the dog], that, the, cat, bit, that, the, man, kicked, barked}
b. * ⟨[the dog] {that, the, cat, bit, that, the, man, kicked, barked}⟩

(B4) a. N = {[the dog], that, the, cat, bit, that, the, man, kicked}
b. ⟨[the dog] {that, the, cat, bit, that, the, man, kicked}⟩
c. ⟨[the dog] ⟨that {the, cat, bit, that, the, man, kicked}⟩⟩
d. * ⟨[the dog] ⟨that ⟨the {cat, bit, that, the, man, kicked}⟩⟩⟩

(B5) a. N = {the, cat}
b. ⟨the {cat}⟩
c. ⟨the ⟨cat { }⟩⟩ => derivational layer

(B6) a. N = {[the dog], that, [the cat], bit, that, the, man, kicked}
b. ⟨[the dog] {that, [the cat], bit, that, the, man, kicked}⟩
c. ⟨[the dog] ⟨that {[the cat], bit, that, the, man, kicked}⟩⟩
d. * ⟨[the dog] ⟨that ⟨[the cat] {bit, that, the, man, kicked}⟩⟩⟩

(B7) a. N = {[the cat], bit}
b. ⟨[the cat] {bit}⟩
c. ⟨[the cat] ⟨bit { }⟩⟩ => derivational layer

(B8) a. N = {[the dog], that, [[the cat] bit], that, the, man, kicked}
b. ⟨[the dog] {that, [[the cat] bit], that, the, man, kicked}⟩
c. ⟨[the dog] ⟨that {[[the cat] bit], that, the, man, kicked}⟩⟩
d. ⟨[the dog] ⟨that ⟨[[the cat] bit] {that, the, man, kicked}⟩⟩⟩
e. ⟨[the dog] ⟨that ⟨[[the cat] bit] ⟨that {the, man, kicked}⟩⟩⟩⟩
f. * ⟨[the dog] ⟨that ⟨[[the cat] bit] ⟨that ⟨the {man, kicked}⟩⟩⟩⟩⟩

(B9) a. N = {the, man}
b. ⟨the {man}⟩
c. ⟨the ⟨man { }⟩⟩ => derivational layer

(B10) a. N = {[the dog], that, [[the cat] bit], that, [the man], kicked}
b. ⟨[the dog] {that, [[the cat] bit], that, [the man], kicked}⟩
c. ⟨[the dog] ⟨that {[[the cat| bit], that, [the man], kicked}⟩⟩
d. ⟨[the dog] ⟨that ⟨[[the cat] bit] {that, [the man], kicked}⟩⟩⟩
e. ⟨[the dog] ⟨that ⟨[[the cat] bit] ⟨that {[the man], kicked}⟩⟩⟩⟩
f. ⟨[the dog] ⟨that ⟨[[the cat] bit] ⟨that ⟨[the man] {kicked}⟩⟩⟩⟩⟩
g. ⟨[the dog] ⟨that ⟨[[the cat] bit] ⟨that ⟨[the man] ⟨kicked { }⟩⟩⟩⟩⟩⟩
=> derivational layer

(B11) a. N = {[[the dog] that [[the cat] bit] that [the man] kicked], barked}
b. ⟨[[the dog] that [[the cat] bit] that [the man] kicked] {barked}⟩
c. ⟨[[the dog] that [[the cat] bit] that [the man] kicked] ⟨barked { }⟩⟩
=> derivational layer

=> 6 derivational layers

C. Type C: 4 layers

(6) c. [[[The dog] **that bit** [**the man**] *that kicked the cat*] barked]

Derivation:

(C1) a. N = {the, dog, that, bit, the, man, that, kicked, the, cat, barked}
b. * ⟨the {dog, that, bit, the, man, that, kicked, the, cat, barked }⟩

(C2) a. N = {the, dog}
b. ⟨the {dog}⟩
c. ⟨the ⟨dog { }⟩⟩ => derivational layer

(C3) a. N = {[the dog], that, bit, the, man, that, kicked, the, cat, barked}
b. * ⟨[the dog] {that, bit, the, man, that, kicked, the, cat, barked}⟩

(C4) a. N = {[the dog], that, bit, the, man, that, kicked, the, cat}
b. ⟨[the dog] {that, bit, the, man, that, kicked, the, cat}⟩
c. ⟨[the dog] ⟨that {bit, the, man, that, kicked, the, cat}⟩⟩
d. ⟨[the dog] ⟨that ⟨bit {the, man, that, kicked, the, cat}⟩⟩⟩
e. * ⟨[the dog] ⟨that ⟨bit ⟨the {man, that, kicked, the, cat}⟩⟩⟩⟩

(C5) a. N = {the, man}
b. ⟨the {man}⟩
c. ⟨the ⟨man { }⟩⟩ => derivational layer

(C6) a. N = {[the dog], that, bit, [the man], that, kicked, the, cat}
b. ⟨[the dog] {that, bit, [the man], that, kicked, the, cat}⟩

c. 〈[the dog] 〈that {bit, [the man] that, kicked, the, cat}〉〉
d. 〈[the dog] 〈that 〈bit {[the man], that, kicked, the, cat}〉〉〉
e. 〈[the dog] 〈that 〈bit 〈[the man] {that, kicked, the, cat}〉〉〉〉
f. 〈[the dog] 〈that 〈bit 〈[the man] 〈that {kicked, the, cat}〉〉〉〉〉
g. 〈[the dog] 〈that 〈bit 〈[the man] 〈that 〈kicked {the, cat}〉〉〉〉〉〉
h. 〈[the dog] 〈that 〈bit 〈[the man] 〈that 〈kicked 〈the {cat}〉〉〉〉〉〉〉
i. 〈[the dog] 〈that 〈bit 〈[the man] 〈that 〈kicked 〈the 〈cat { }〉〉〉〉〉〉〉〉

=> derivational layer

(C7) a. N = {[[the dog] that bit [the man] that kicked the cat], barked}
b. 〈[[the dog] that bit [the man] that kicked the cat] {barked}〉
c. 〈[[the dog] that bit [the man] that kicked the cat] 〈barked { }〉〉

=> derivational layer

=> 4 derivational layers

References

Abelson, Harold & Gerald J. Sussman. 1984. *Structure and Interpretation of Computer Programs*. Cambridge, MA: MIT Press.

Arsenijević, Boban & Wolfram Hinzen. 2010. Recursion as a human universal and as a primitive. *Biolinguistics* 4, 165–173.

Bader, Markus. 2011. Complex center-embedded relative clauses in German. Ms. University of Konstanz.

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

Chomsky, Noam. 2008. On phases. In Robert Freidin, Carlos P. Otero & Maria L. Zubizarreta (eds.), *Foundational Issues in Linguistic Theory: Essays in Honor of Jean-Roger Vergnaud*, 133–166. Cambridge, MA: MIT Press.

den Dikken, Marcel. 2007. Phase extension: Contours of a theory of the role of head movement in phrasal extraction. *Theoretical Linguistics* 33, 1–42.

Everett, Daniel L. 2005. Cultural constraints on grammar and cognition in Pirahã: Another look at the design features of human language. *Current Anthropology* 46, 621–646.

Gallego, Ángel J. 2010. *Phase Theory*. Amsterdam: John Benjamins.

Gallego, Ángel J. & Juan Uriagereka. 2007. Sub-extraction from subjects: A phase theory account. In José Camacho, Nydia Flores-Ferrán, Liliana Sánchez, Viviane Déprez & María José Cabrera (eds.), *Romance Linguistics 2006*, 149–162. Amsterdam: John Benjamins.

Hinzen, Wolfram & Boban Arsenijević. To appear. On the absence of X-within-X recursion in human grammar. *Linguistic Inquiry*.

Kayne, Richard S. 1994. *The Antisymmetry of Syntax*. Cambridge, MA: MIT Press.

Nevins, Andrew, David Pesetsky & Cilene Rodrigues. 2009. Pirahã exceptionality: A reassessment. *Language* 85, 355–404.

Phillips, Colin. 1996. *Order and Structure*. Cambridge, MA: MIT dissertation.

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

Shannon, Thomas F. 1992. Toward an adequate characterization of relative clause extraposition. In Irmengard Rauch, Gerald F. Carr & Robert L. Kyes (eds.), *On Germanic Linguistics: Issues and Methods,* 253–281. Berlin: Mouton de Gruyter.

Svenonius, Peter. 2001. On object shift, scrambling, and the PIC. *MIT Working Papers in Linguistics* 39, 267–289.

Trotzke, Andreas, Lyn Frazier & Markus Bader. Submitted. The performance interface in language design: Two case studies. In Anna Maria Di Sciullo (ed.), *Biolinguistic Perspectives on Language Design*. Cambridge, MA: MIT Press.

Uriagereka, Juan. 1999. Multiple Spell-Out. In Samuel D. Epstein & Norbert Hornstein (eds.), *Working Minimalism,* 251–282. Cambridge, MA: MIT Press.

Uszkoreit, Hans, Thorsten Brants, Denys Duchier, Brigitte Krenn, Lars Konieczny, Stefan Oepen & Wojciech Skut. 1998. Studien zur performanzorientierten Linguistik: Aspekte der Relativsatzextraposition im Deutschen. *Kognitionswissenschaft* 7, 129–133.

Weinberg, Amy. 1999. A minimalist theory of human sentence processing. In Samuel D. Epstein & Norbert Hornstein (eds.), *Working Minimalism,* 283–315. Cambridge, MA: MIT Press.

Zwart, Jan-Wouter. 2009. Prospects for top-down derivation. *Catalan Journal of Linguistics* 8, 161–187.

Zwart, Jan-Wouter. 2011a. Recursion in language: A layered-derivation approach. *Biolinguistics* 5, 43–56.

Zwart, Jan-Wouter. 2011b. Structure and order: Asymmetric merge. In Cedric Boeckx (ed.), *The Oxford Handbook of Linguistic Minimalism,* 96–118. Oxford: Oxford University Press.

Andreas Trotzke
Universität Konstanz
Fachbereich Sprachwissenschaft
Universitätsstraße 10
78457 Konstanz
Germany
andreas.trotzke@uni-konstanz.de

Antje Lahne
Universität Konstanz
Fachbereich Sprachwissenschaft
Universitätsstraße 10
78457 Konstanz
Germany
antje.lahne@uni-konstanz.de