Some problems with merge

Michael Brody

UCL

Abstract

I argue that in the standard merge-based system, where move is a subcase of merge, both allowing and disallowing sideward movement appears to lead to the conclusion that movement is not part of syntax. I propose that we can nevertheless save the idea that move is a subcase of the syntactic structure building rule, --which appears necessary if we wish to retain move as a (non-stipulated) syntactic operation. To do this we seem to need an alternative theory of syntactic assembly that provides a principled way of ruling out the (equivalent of) sideward movement and sideward intervention.

1 Introduction

Uncontroversially, the structure of natural language sentences is complex enough to include categories that immediately dominate more than one non-terminal category, more than one category with internal structure. The standard assumption is that sentence structure assembly is accomplished by merge, an operation that joins two categories to create a third. Given merge-based assembly, categories that immediately contain two non-terminals can only be derived through the merger of the products of initially independent sub-derivations producing these non-terminals.

Given the further assumption that move is merger of a previously merged category, the question arises if this remerge operation, like other cases of merge, also has the ability to link independent sub-derivations; --whether we can move an element from a sub-derivation A to another sub-derivation B at a point where A and B are not yet merged, and are independent from each other. Such a derivational step is usually referred to as sideward movement. I will argue in section 2 that if sideward movement exists, then it effectively voids movement of its core properties of locality and c-command. Movement without its characteristic properties would be a pointless operation that furthermore carried a significant unjustifiable cost relating to its derivational memory requirement. So sideward movement leads to a contradiction: if sideward movement exists then movement, and therefore also of course sideward movement, does not.

In section 3 I argue that given the answer to the question of why move exists, --that move is the subcase of merge, where merge applies to a previously merged object--, the assumption that sideward movement exists is difficult to avoid.

I will experiment with an understanding of the mechanics of merge that in the context of the extension condition entails the impossibility of sideward movement. But I will have to conclude that mainly due to various problems with the extension condition, this solution is not satisfactory. Perhaps the most important of these problems is that move obeys a more general c-command condition than the one the extension condition is capable of ensuring. Move not only targets c-commanding positions, but it also ignores non-c-commanding

interveners. Move does not even "see" sideward. In the standard merge-remerge setting there appears to be little hope of finding a non-stipulative way of excluding sideward movement, and even less of finding an explanation of this more general c-command condition, of "sideward blindness".

We could of course directly stipulate the impossibility of sideward movement in various ways. But such stipulations, making move and merge distinct (in their ability to span sub-derivations) at least implicitly question the thesis that move is remerge, and consequently also the justification for the status of move as a syntactic operation.

Thus, in the merge-based system both allowing and disallowing sideward movement appears to lead to the conclusion that movement does not exist as a syntactic operation. I propose in section 4 that we can nevertheless save the idea that move is a subcase of the syntactic structure building rule, --which is necessary to retain move as a non-stipulated syntactic operation. To do this we need an alternative theory of syntactic assembly without merge (and remerge) that does not allow the (equivalent of) sideward movement and sideward intervention.

I adopt the approach where syntactic phases, linked only at the interfaces, are paths from the initial symbol to the terminal elements, and that the syntactic assembly operation is concatenate rather than merge (Brody 2015, 2017, 2018). Among various advantages, the c-command requirement of reconcatenate (the correspondent of remerge in this approach), and the principled lack of (the equivalent of) sideward movement and sideward intervention relations immediately follow.

2 Islands and c-command

2.1 Islands

2.1.1 Successive cyclic/step sideward movement

In order exemplify the problems that arise from the assumption that sideward movement is possible, let us turn to Nunes (1995, 2001) proposal that certain structures should have a derivation involving this type of movement. The syntactic existence of most of these structures, -head movement, post-cyclic LF movement, post-cyclic late insertion of adjuncts- is at least highly controversial, with theoretically more satisfactory and empirically more promising alternatives being available. I will therefore exemplify the problems that arise by discussing his strongest proposal, that of deriving sentences containing parasitic gaps via sideward movement. (Most of these comments will apply mutatis mutandis to the other constructions if these turn out to exist in syntax.)

Nunes (2001) suggests that "the derivation of [(1)] does not exhibit a CED effect because *which paper* moved [from the pg to the t position in (1)] prior to the derivational step where the gerundive clause became an island; therefore it was free to undergo additional movement, regardless of the status of the gerundive clause later in the derivation" (p.328).

(1) Which book did you file t [without reading pg]

So let us suppose that sideward movement from an island constituent, like the gerundive clause in (1) is possible before the island constituent merged with the rest of the structure. In other words, a constituent becomes an island only when it merges with the rest of the structure. The immediate question that arises is why are islands ever islands, why cannot all islands be similarly escaped?

Nunes notes that a difference between the parasitic gap structure in (1) and the ungrammatical extraction in (2) below is that in (1), unlike in (2) movement of the wh-phrase to its final landing position is from outside the CED island. "The crucial difference between these structures is that in (1), the *wh*-copy in the object of *file* is clearly not inside any island…"(p.328).

(2) *Which book did you review this paper [without reading t]

But while there is indeed such a difference, this does not address the problem that his analysis of (1) raises. If an island becomes an island only when it merges with the rest of the structure and therefore movement from an island-to-be is possible before the island joins the rest of the structure then why cannot the *wh*-phrase escape from inside the CED-island gerundive in (2) before the gerundive merges with the rest of the structure?

Cyclicity, whether or not a consequence of the extension condition, might ensure that (2) cannot be derived directly, since the adjunction site of the gerundive is lower in the tree than the ultimate landing site of the *wh*-phrase. So the gerundive will already be attached and therefore an island by the time the *wh*-phrase moves to sentence initial position. But given the option of successive movement via intermediate sites, the *wh*-phrase can attach first to the VP to which the *without*-phrase attaches, but lower than the *without*-gerundive and therefore before this phrase attached to the VP and became an island:

(3) *Which book did you [[t[criticize John]] [before reading t]]

It is perhaps to rule out such intermediate *wh*-copies among other structures that Nunes and Uriagereka (2000) stipulate that "… Last Resort imposes that the extra copy be legitimated, which separates instances where this copy is made with no purpose other than escaping an island (a CED effect) from instances where the copy is made in order to satisfy a theta-relation (a parasitic gap construction)" (p.41). There is no obvious way of making this constraint relevant only to sideward movement, so if last resort is to rule out (3), then the proposal entails the en bloc rejection of the successive step (whether phase based or not) theory of locality, without offering an alternative.

But even going along with this and assuming that *wh*-movement never involves intermediate positions that would not directly trigger movement by themselves, would not help. Sideward movement of the *wh*-phrase in (2) to the matrix VP could have been 'legitimated' for example by a Case position as in (4), or a thematic specifier of v position as in (5), (6), both of which are lower in the tree than the adjunct island. In (4) sideward movement out of the adjunct island targets the accusative Case position of *believe* normally occupied by the raised subject of an embedded infinitive:

(4) **Which book did you [[believe t ~~believe~~ that S][before reading/being read t]]

The parallel example where sideward movement proceeded through the matrix v-spec position as in (5) might well be ruled out by whatever generally excludes parasitic gap structures with a nominative subject real gap. But this consideration would not help with (6):

(5) **Which lamp has [[t v the ice melted] [before you bought t]]

(6) **Which lamp did you believe t ~~believe~~ to have [[t v the ice melted] [before you bought t]]

On the sideward movement theory, just like (4), examples like (6) should be grammatical extractions. Structures like (6) involving raising of the lower subject in the real gap position to the accusative of the matrix clause (followed by verb raising across the raised subject) normally license parasitic gaps, eg: “Which book did you [[believe t ~~believe~~ t to have been on sale][before deciding to read t]]”. Note also that without sideward movement (4) and (6) are unproblematically ruled out as parasitic gap structures, due to the higher (‘real’) gap of the wh-phrase not originating in a VP-internal theta position.

There are many other potential ‘legitimizing’ positions in the VP, providing potential escape hatches from islands if sideward movement is possible. Case and v are special among them only in that they are clearly in a lower position than the VP adjunct containing the gap in (1)-(6). But other legitimate midfield landing positions in various languages, like for example focus in Hungarian or various scrambling positions in German or the landing position of heavy NP shift in English, even if higher than this type of VP adjunct are clearly lower than the subject of the sentence. Hence sideward movement to any of these before the subject is merged with the rest of the clause will incorrectly result in an escape from the subject island. It does not seem promising to try to attribute incorrect island escaping sideward movement cases to some version of the last resort condition, whether or not this condition is meant to hold for movement in general or somehow restricted to constrain only sideward movement.

2.1.2 Parasitic gaps in multiple islands and cyclicity

If we have a derivation that can escape an island, then iterating the steps, we can escape any number of islands. But in fact, given sideward movement we can escape multiple islands even in one step.

Consider the ungrammatical parasitic gap structure in (7) where the parasitic gap is inside a second island:

(7) *Which book did you finally read t after leaving the bookstore without finding pg?

Nunes, and Nunes and Uriagereka assume that some version of cyclicity will ensure that movement from pg to t is not possible. They assume that merger of *read* with its object must occur later than the point when the lower *without*-clause is attached and becomes an island. To attribute this to cyclicity seems to be a dubious move, since merger of *read* with its object takes place in a sub-derivation that is independent from the sub-derivation that assembles the adjunct clause(s). The two sub-derivations are independent at least until the *after*-clause merges with the matrix VP. In consequence, the standard extension condition, which does

not pertain to the question of how the operations of a sub-derivations are ordered with respect to the operations of another independent sub-derivation cannot ensure that sideward movement from the pg position in the double island to the object of *read* is ruled out.

In addition sideward movement from the object of *read* position to the pg position inside the double island must also be ruled out. Addressing this problem, Nunes and Uriagereka propose to rule out sideward movement into embedded domains by assuming that phases have separate numeration subarrays and a derivation is illicit if "it accesses a new subarray before it has used up the lexical items of the active subarray."(p.40.) Let us call this condition subarray cyclicity. Subarray cyclicity must be additional to the extension condition, hence it destroys the idea that some computationally natural extension condition is the cause and the full explanation of the cyclic nature of derivations. Nunes and Uriagereka do not discuss in detail how they expect these principles to work, but they state that they assume that in (7) *after* is a member of the subarray of the phase that includes the object of *read*. We might suppose then, that movement to a position inside the *after* clause from the object of *read* position cannot be attempted until *after,* and therefore until the *after*-clause of which *after* is part, has been merged and became an island, -thus preventing the movement in question. The account appears to predict incorrectly that the adjunct clause will cease to be an island for this type of movement if it is not introduced by a lexical element (*Which book did you finally read t leaving the bookstore without finding pg?) and does not even address the problem of sideward movement in the opposite direction, discussed in the previous paragraph.

We cannot solve the problems of (rightward and leftward) multiple island violations with any generality by trying to find some better way to ensure that sideward movement can proceed only from a phase to an immediately superordinate phase at least if we wish to retain the relevance of sideward movement to the parasitic gap construction. The parasitic gap and the real gap are typically in separate sub-derivations, but as (8) exemplifies, they can furthermore each be in a phase that is neither superordinate nor subordinate to the other.

(8) Which book did everyone who bought pg want his classmates to read t

If we wish to derive parasitic gap structures via sideward movement, then it is impossible to rule out sideward movement from multiple islands with any restriction along the lines of subarray cyclicity. As we have just seen, in (8) the pg is in a phase P in a sub-derivation SD that is initially independent from the sub-derivation SD' that contains the phase P' containing t. But under a sideward movement approach to parasitic gaps, movement from pg to t must be possible, since the construction is grammatical.

Furthermore, even if structures like (8) were ungrammatical, we could not use a condition like subarray cyclicity to prevent ungrammatical island violations of the sub-derivation crossing sideward movement from pg to t in general. Consider such a movement in (9). Since the two relevant phases containing the pg and the real gap are in initially independent sub-derivations, and these phases will not be sub-/super-ordinated to each other, both

cyclicity of sub-derivations (the extension condition) and the putative cyclicity of subarrays are irrelevant:

(9) *Which book did everyone who met a woman who bought pg want his classmates to read t

Extending cyclicity to how super- and subordinate phases or sub-arrays of numerations are accessed is incapable of regulating the order of operations in a phase P relative to the order of operations in another phase P' where P and P' are in independent sub-derivations and are not destined to be sub-/super-ordinated to each-other.

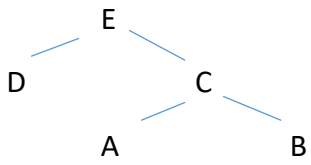
To summarize, allowing sideward movement appears to void island constraints in multiple ways. It would appear that in a theory with sideward movement, obeying islands conditions will not be a property of the movement/remerge operation.

2.2 C-command

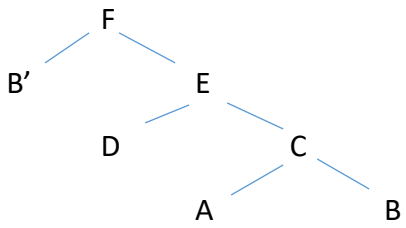
The c-command requirement on syntactic movement is often attributed the extension condition, the requirement that merger operations (merge and its subcase, move or remerge) extend the tree built by earlier derivational steps.

In (10a) remerging B with E creating (10b) obeys the extension condition, since the condition allows accessing B in E and remerging B with E does not modify E internally. The operation extends the tree rooted in E, by adding B' and F externally to E. Consequently c-command will hold between the two copies of B.

(10a)

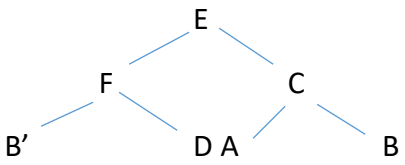


(10b)



Remerging B with D in (10a), resulting in (10c) on the other hand would not extend the tree rooted in E, built earlier, since B' and F are now added internal to E. If this is not allowed, then the remerged category will necessarily c-command its originally merged site.

(10c)



However, in a merge-based derivational approach sideward movement provides another way of deriving (10c), one that does not violate the extension condition. B' can be merged with D before F, the category immediately containing B' and D is merged with C.

The existence of possibly legitimate derivations that produce structures like (10c) entails that the c-command requirement on movement structures cannot possibly be identical to, or be a trivial consequence of, the extension condition.

Nunes is aware of this consequence of sideward movement and suggests that the c-command requirement is a condition not on movement but on LF chains.

As we have seen earlier, given sideward movement, island conditions similarly cannot be conditions on movement, presumably then they are also conditions on chains. However, if the core individuating properties of movement are not taken to be properties of movement then movement becomes unnecessary and therefore redundant in syntax. One might then assume that deplacement, c-command and locality do not form a natural cluster (as e.g. in Koster 1987) or that they do but characterize some other relation, like chains (as e.g. in Brody 1995)

Nunes' proposal attributing the c-command condition to LF chains seems more reasonable to me than Hornstein's (2009) later approach, which sticks more closely to the letter, though as far as I can see not to the spirit of the minimalist approach, and envisions a gigantic accidental conspiracy to ensure c-command:

"If grammatical dependencies are coded via Merge, if Move is Copy plus Merge (or Merge/Remerge), if computations are monotonic increasing (i.e. obey Extension), if grammars optimize by preferring shorter relations to longer ones, if grammars use Boolean resources, if grammatical operations are last resort (deterministic), if grammars segregate theta, case and A'-domains, then c-command will figure dominantly in grammatical processes." (p. 52.) Much of this seems problematic, in part for the reasons discussed in 2.1 above. But even if the argumentation was unassailable, and even if we suspended disbelief in taking c-command to be the consequence of some cosmic computational accident, nothing in these assumptions or their consequences would affect the conclusion above that given sideward movement, syntactic movement remains without properties and therefore unmotivated as a syntactic entity.

Perhaps generally, but at least in the merge based framework, there seems to be only one reasonable potential way to escape the conclusion that the relation obeying island conditions and c-command, the relation standardly taken to be movement, is not part of syntax. This is to ensure that sideward movement does not exist. Without sideward movement, movement operations won't be able to cross nodes that are future island

boundaries before these boundaries are formed. And they won't be able to land in a non c-commanding position in violation of the extension condition.

3 Attempts to exclude sideward movement and sideward blindness

3.1 Conditions on remerge

We might try to exclude derivations involving sideward movement in general by prohibiting movement across disconnected sub-derivations. Within a single sub-derivation the extension condition will ensure that movement must land in a c-commanding position. But this seems to be a step in a wrong direction. Remerge is the same operation as merge, which we know must be able to link independent sub-derivations. Attributing different properties to move and to merge weakens the hypothesis that they are identical. In this way we get close to the obviously undesirable early minimalist suggestion that move is an imperfection of syntax. As was already clear at the time, attributing the relations captured by move to the interpretative component where a relevant concept of identity is independently necessary is at least a likely improvement over the syntactic imperfection hypothesis. It follows that stipulating the nonexistence of sideward movement by attributing different properties to merge and remerge effectively questions the existence of syntactic movement. Proceeding this way we reach an impasse, both assuming sideward movement and prohibiting it leads to questioning the existence of syntactic deplacement operations.

Would it help to say that sideward movement creates a contradiction: within the sub-derivation targeted by sideward movement the merger appears to be a case of external merge while taking into account the launching sub-derivation we see a case of internal merge. Clearly if remerge is simply a subcase of merge, then there is no reason why this ambiguity of the two possible viewpoints should create a contradiction that crashes the derivation. It is only if external and internal merge are different operations that it becomes possible to prohibit their overlap. So the core problem with this approach will be the same as with the direct prohibition of sideward movement: the rejection of the hypothesis that move is a subcase of merge.

Another suggestion in the literature as a potential way to exclude sideward movement is to assume that move must land in a c-commanding position, perhaps because move is dependent on Agree, --and Agree operates under c-command. Apart from the fact that this loses the extension based explanation of the c-command property of movement relations, yet again the same problem arises as before. With move but not external merge being dependent on Agree, move and merge drift apart, leading to a syntactic movement rule of questionable status.

3.2 Merge as copy

Within a single sub-derivation sideward movement, that is, movement to a non c-commanding position is excluded by the extension condition. It may be natural to try to use

the same condition to rule out also sideward movement across sub-derivations. This might be possible if it turned out that sideward movement necessarily involved modifying previously built structure. We could engineer this by assuming that merge operates by joining copies of its input categories, subsequently deleting the originals. So suppose Merge deletes its input. In the cases of joining elements from the numeration or joining sub-derivations with each other this is uneventful. In the case of sub-derivation internal remerge we might wonder if deletion of the original of the moved element violates the extension condition, since such an operation would modify structure already built. But here the requirement that the original of the moved element be deleted can be satisfied vacuously, since it will have been deleted when deletion applied to the other input of merge, the sub-derivation being extended by remerge. (At least if the option is taken where deletion of the sub-derivation has applied by the time deletion of the original of the moved element is considered.) In the case of sideward movement however the launching site of movement is in a sub-derivation that is distinct from the sub-derivation that merges with the moved element. Hence deletion of the movement target sub-derivation leaves the original of the moved element undeleted and merge can only delete this category in violation of the extension condition.

(11) a. merge: A , B → (Acopy +Bcopy), A→0, B→0

b. (sub-derivation internal) remerge: $(_{B}\ldots A\ldots)\rightarrow(\text{Acopy} +(_{\text{Bcopy}}\ldots\text{Acopy}\ldots))$, $(_{B}\ldots A\ldots)\rightarrow 0$, A→0 (vacuously)

c. sideward remerge: C, $(_{B}\ldots A\ldots)\rightarrow(\text{Ccopy}+\text{Acopy})$, C→0, A→0 i.e. $(_{B}\ldots A\ldots)\rightarrow(_{B}\ldots 0\ldots)$

While this approach seems better than the ones that stipulate distinctions between merge and remerge, it still makes the arbitrary assumption that prohibition of deletion of the original moved element by the extension condition does not simply prevent this deletion but also results in a crashed derivation.

Further problems arise with the extension condition that this account crucially assumes. On a slightly closer examination the extension condition turns out to be a less than optimal principle. The extension condition does not seem to be derivable from an interpretive monotony requirement. If interpretation of structures generated by merge is phase by phase or even rule by rule, modifying an already interpreted substructure should not affect the interpretation. If only the fully formed sentential structure is subject to interpretation, then interpretation is not going to have access to the information of whether earlier derivational steps have been changed. So the rationale of the extension condition must indeed be computational simplicity. But for such a condition it is strikingly unambitious. A natural condition to simplify computation in a derivational framework would be one that requires derivational operations to have no access at all to earlier stages of the derivation. Call this the full extension condition. The full extension condition would strongly restrict derivations ensuring that derivational steps cannot have access to (a representational memory of) earlier derivational steps. As a consequence derivational steps cannot then modify the results of earlier steps either and must therefore extend the structure, -the standard extension condition.

The full extension condition is of course incompatible with the assumption that syntactic move/remerge exists, since this operation necessarily needs access to the earlier merged copy or merge position of the moved element. Given syntactic move, what remains is a rather dubious stipulative residue of the full extension condition, the standard extension condition, according to which earlier stages of the derivation can be accessed, but cannot be modified.

3.3 Sideward blindness

Whether or not we engineer a mechanism for merge under which the c-command property of movement entails the impossibility of sideward movement, the explanatory value of attributing c-command to the extension condition, while technically feasible, appears to be minimal. There is also a further, perhaps even stronger consideration that suggests that the extension condition will not provide the solution. The requirement that movement must land in a c-commanding position is only a subcase of a more general c-command condition on movement, one that clearly cannot be a consequence of the extension requirement. Call this more general condition sideward blindness:

(12) sideward blindness: movement is insensitive to positions that neither dominate nor c-command the launching site

It appears that movement must land in a c-commanding position because more embedded positions are not visible for it, as shown by the fact that potential interveners function as actual interveners only when they c-command the movement launching site:

(13) a. *I wonder what who saw t

b. I wonder what [the man who just called us] saw t?

(14) a. ?*What did you ask who bought t

b. What did you say [the man who just called us] bought t?

c. What did you think [the question of who called us] really meant t?

In contrast with (13a), there is no superiority violation in (13b) where the potential intervener does not c-command. Similarly (14b) and (14c) in contrast with (14a) show no *wh*-island violation.

It seems unsatisfactory to account for the fact that potential interveners only become effective in positions that c-command the movement site separately from the requirement that the landing position of movement must c-command the launching site. These seem to be two aspects of a single property, accounting for them separately would miss the generalization in (12).

Hornstein (2009) exemplifies an approach that misses the generalization in (12). He suggests that the fact that only c-commanding interveners count is due to a shortest path requirement where shortest path is understood in terms of subset relations. The path from

the target of movement to the c-commanding, but not to the non c-commanding, intervener is set of nodes that is a subset of the set of nodes that is the path from the target to the movement launching site. This makes the c-command condition on interveners unrelated to the c-command condition on landing sites, a part of his conspiracy theoretical account of c-command quoted in section 2.2 above. Note also that nothing in principle prevents abstracting away from labels, if these exist at all in the first place, and compare also the length of paths in the cases of non c-commanding interveners by checking subset relations between sets of (unlabeled, hence identical) categories. (Confusingly, given that integers can be thought of as a matter of subset comparisons, Hornstein attributes the assumed necessity of subset comparison of paths to the assumption that syntax cannot count.)

The concept of path seems relevant however. Movement can land in any position that is sister to a target node that is on the path between the launching site and the initial symbol of the sentence. This suggests that perhaps interveners that c-command the launching site are not the actual interveners that are visible for movement either, it is rather their agreeing sister node on the path between the launching site and the target of the movement that blocks the operation. This would provide a partial unification with CED type islands, where no obvious c-commanding interveners exist. If so, then we can restate the sideward blindness condition in a stricter and simpler form, to refer only to dominating nodes:

(15) strict sideward blindness: movement is insensitive to non-dominating positions

To summarize this section, it seems fair to conclude that we have not found a theoretically satisfactory way of ruling out sideward movement within the standard minimalist merge-remerge system. It seems impossible to achieve this without conditions which either weaken the hypothesis that move/remerge is a subcase of merge and consequently question the status of move/remerge as a syntactic operation, -- or rely on stipulations with little explanatory depth. Furthermore none of these ways of excluding sideward movement helps to capture the core generalization expressed by the sideward blindness condition.

4 One-dimensional syntax and c-command

Recall the central problem. The idea that move is remerge, that is, simply an application of merge, and the assumption that merge is capable of applying to two independent sub-derivations together appear to entail that remerge can also link independent sub-derivations. This means that sideward movement is part of the grammar. And this in turn causes move to have no syntactic properties, or essentially equivalently, causes move not to be a syntactic operation.

The alternative seems to be to reject the idea that move is remerge. But this leads to the same conclusion of move-free syntax, at least as long as we cannot think of an alternative way of conceptualizing movement not as an ‘imperfection’ but part of the minimally necessary syntactic equipment to link the interfaces.

But there is yet another alternative, and this is what I propose to attempt here:
we might try to preserve the hypothesis that move is a subcase of merge but reject the

assumption that merge joins sub-derivations. We could try at first to take all sub-derivations to be separate phases, to be linked only at the interface. This approach immediately sinks due to the fact that it cannot allow movement from a sub-derivation, like for example a complex specifier, to an ultimately c-commanding position located outside the sub-derivation of this specifier. But valid movement operations are generally able to cross sub-derivation boundaries in this way.

Movement does not care about the sub-derivations of the merge-based structure building, it can land in any local c-commanding position. Sidestepping the question of locality here, movement can land in any position that c-commands the launching site. In other words it can land in any position that is sister to a target node that is on the path between the launching site and the initial symbol of the sentence.

Suppose then that the domain of syntactic rules is the path from a terminal to the initial symbol, the one-dimensional syntax program (Brody 2015). We might think paths as phases, of a kind that are assembled into a sentence structure only at the interfaces. Call these one-dimensional syntactic structures, each a linear order from the initial symbol to a terminal, p-strings. Accordingly, the operation that assembles p-strings is not merge but concatenate.

If interpretive rules for a natural language sentence take as input a structure created by syntax, then they need a set of trees, and syntax, by definition, must provide this set. It does not follow however that syntax must build trees. If syntax builds segments of trees (phases) then the segments that syntax builds may not be trees, if together they form a tree. As we have seen in previous sections above, trees seem too permissive as syntactic objects, they are structures that lead us to expect the existence of sideward relations.

Returning to p-strings, one way of instantiating the program of one-dimensional syntax is to assume that there are certain (not necessarily proper) path segments, the universal hierarchies of functional sequences, which are provided for syntax by systems external to it, presumably by lexical and other interface systems. Call these path segments c-strings. C-strings are the building blocks for the assembly of p-strings. C-strings always end with a terminal element.

Assume further that the syntactic operation of concatenate can join an initial p-string segment to a final p-string segment. A final p-string segment is one that ends with a terminal, hence c-strings are final p-string segments. For example in the sentence “John left” the verb comes with a partially or fully ordered series of functional projections, an instantiation of which is a c-string. Concatenating this with an initial symbol we get a p-string, P1, leading from the initial symbol to the verb. Concatenate can then take the initial segment of this string, down to the node, say T, where subjects are standardly attached, and concatenate that segment with the nominal c-string terminating with *John*, creating another p-string, P2, that connects the top of the sentence to *John.* If we have “the man left” instead of “John left”, then P2 leads to *man* and we need to construct a third p-string by concatenating an initial segment of P2 with a c-string terminating with the article.

Crucially, the domain of each operation of concatenate is only a single p-string. Syntax is one dimensional: both the domain and the range of concatenate is a set of one-dimensional

objects. Assembly of p-strings into the tree structure of the sentence can take place only at the interface. This is based on identity relations between nodes. Take the illustration “John left” again. The initial substrings of P1 and P2, from the initial symbol to the node where P2 was concatenated with P1, are identical. The interpretive component, which unlike syntax has the ability of processing more than one p-string at a time, will therefore see the two p-strings as a branching tree.

Returning now to movement, in analogy with remerge we might take this to be the case where concatenate applies to an already concatenated object. Recall that concatenate joins initial and final string segments. So in addition to adding a c-string (a final segment) to the initial segment of the p-string, concatenate will have the reconcatenate option of creating a new p-string by adding the final segment, FS, of a p-string P to its initial segment.

Since syntax is one-dimensional and (re)concatenate can apply only to a single p-string at a time, it follows that reconcatenate will not be able to connect this FS to any position that is not part of P. This allows the highest node of FS (HNFS) to be attached only to a node that either precedes HNFS or to one that HNFS precedes in P. In either case, in the interpretive component there will be a c-command relation between the launching and the landing site of the movement.

But the latter case, attaching to a node that HNFS precedes in P, would attempt to create a string P’ in which the HNFS (originally of P) and any other node between the two positions of HNFS in P’ precedes itself and all other nodes between the two positions of HNFS in P’. In (16b) a final segment of P in (16a) headed by HNFS concatenates with an initial segment of P ending in E:

(16) a. P A B HNFS C D E F

b. P’ A B HNFS C D E HNFS C D E F

But given our central restrictive hypothesis that syntax can only create linear orders, this is impossible. P’ is not a possible p-string. Consequently, in the p-string that results from reconcatenation HNFS must be immediately dominated by a node that dominates its source position in P, and therefore at the interface the reconcatenated (HN)FS will necessarily always c-command the original (HN)FS.

It follows that in one-dimensional syntax the extension condition is also unnecessary in the explanation of c-command (of remerge/reconcatenate). This is a welcome consequence, since as we have seen the extension condition, allowing access to, but not change of, earlier processed domains, is far from an elegant, theoretically desirable addition.

To summarize, the structure of natural language sentences is rich enough to include constituents immediately dominating more than one complex constituents. This makes it unlikely that we can simultaneously maintain the hypothesis that sentence structures are assembled by the operation of merge and the claim that move is remerge, a subcase of merge. The conjunction of these assumptions leads to some version of sideward movement,

which empties the concept of movement of content by voiding it of its core properties of locality and c-command. One possibility is to continue to use merge for syntactic sentence assembly and eliminate movement from syntax, perhaps by resurrecting earlier approaches that locate this relation in the interpretive component. Another possibility is to reject merge, and use concatenate instead to assemble p-strings. This approach is able to preserve the idea that move is simply a subcase of the general assembly operation without having to face the problems of the merge based theory. It also provides an account that at last shows some promise to be an explanation of the c-command property of movement.

References

Brody, Michael. 1995. Lexico-Logical Form: A Radically Minimalist Theory. Cambridge, MA: MIT Press.

Brody, Michael. 1997. Perfect Chains. In Liliane Haegeman ed. Elements of Grammar. Kluwer. pp. 139-167

Brody, Michael. 1998. The Minimalist Program and a Perfect Syntax. In Mind and Language 13: 205-214.

Brody, Michael 2015. One-dimensional syntax. http://ling.auf.net/lingbuzz/002863

Brody, Michael 2017. Two advantages of Precedence Syntax. In Bánréti et al (eds.): Boundaries Crossed: Studies at the Crossroads of morphosyntax, phonology, pragmatics, and semantics. http://ling.auf.net/lingbuzz/003761

Brody, Michael 2018. 'Movement', precedence, c-command. http://ling.auf.net/lingbuzz/004044

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

Chomsky, Noam. 2000. Minimalist Inquiries: The Framework. In Martin et al eds. Step-by-step—Essays on Minimalist Syntax in Honor of Howard Lasnik, Cambridge, MA: MIT Press.

Chomsky, Noam. 2004. "Beyond Explanatory Adequacy" in Adriana Belletti ed. Structures and beyond. Oxford University Press. New York.

Hornstein, Norbert. 2009. A Theory of Syntax: Minimal Operations and Universal Grammar. Cambridge: CUP.

Koster, Jan. 1987. Domains and dynasties: the radical autonomy of syntax. Dordrecht: Foris.

Nunes, Jairo. 1995. The copy theory of movement and linearization of chains in the Minimalist Program. Doctoral dissertation, University of Maryland, College Park

Nunes, Jairo. 2001. Sideward Movement. Linguistic Inquiry 32:2:303-344.

Nunes, Jairo and Juan Uriagereka 2000. Cyclicity and Extraction Domains. Syntax 3:1:20-43

Starke, Michal. 2001. Move Dissolves into Merge: a Theory of Locality. Doctoral Dissertation. University of Geneva