Reducing computation at the interface with the Sensory-Motor Systems
A derivational approach to (Chain) linearization

Gerardo Fernández-Salgueiro

In this paper I develop a derivational approach to linearization under a (version of the) level-free derivational approach to syntactic relations (Epstein et al. 1998; Epstein & Seely 2002, 2006). The approach I propose here differs from most approaches to linearization in that it does not rely on Kayne's (1994) Linear Correspondence Axiom (LCA), but rather on a more intrinsic property of the derivation, i.e., *order of Merge operations*. I will also illustrate some consequences of this approach for the linearization of chains, by taking Nunes's (1999, 2004) theory as a starting point.

1. Goals and outline

Most approaches to linearization in early minimalism (e.g., Chomsky 1995; Guimarães 1998; Nunes 1999, 2004; Richards 2001) rely on Kayne's (1994) notion of asymmetric c-command, a syntactic relation that is defined based on the representation of a tree structure, not on the step-by-step nature of the derivation. These approaches, therefore, independently of their internal coherence and/or predictive power, cannot be considered derivational, because they make use of Kayne's Linear Correspondence Axiom, which is clearly a representational notion (since it is in turn based on asymmetric c-command).[1] Richards (2001:2), for example, writes:

> "I will assume that linearization is accomplished via Kayne's (1994) LCA. Spell-Out considers the set **A** of pairs of asymmetrically c-commanding XPs and X^0s *in the tree which the syntax gives it*, and generates from this a set of instructions for linearization." (my emphasis, GF-S)

Similarly, Nunes (2004: 44), writes:

> "[…] the operation *Linearize*, which I take to be the procedure that maps *a given syntactic structure into a sequence of terminals*, in compliance with the LCA." (my emphasis, GF-S)

[1] Notice that this is not a critique of Kayne's LCA itself, which is perfectly legitimate under a representational approach.

As can be seen in these two quotes, it is assumed that Spell-Out/Linearize can generate a sequence of terminals by inspecting a syntactic tree and applying the LCA to it.

Another good example of how representational notions are used for linearization purposes is the following paragraph by Chomsky (1995: 340):

> "The lowest Z that dominates *q* and m_1 is L, which also dominates [m_2, m_1]. Therefore, *q* and [m_2, m_1] asymmetrically c-command *r* and S, however we interpret "disconnected". What are the c-command relations within [m_2, m_1]? As noted, m_1 does not c-command anything."

In sum, these approaches crucially depend on a *representation* of the hierarchical structure of a sentence.

More recently, there have also been several attempts to build derivational models of linearization, like Epstein et al. (1998) (see also Epstein and Seely 2002); Uriagereka 1999 and Pesetsky & Fox (to appear), which I discuss below.[2] Following this derivational trend, I am going to argue in this paper that the derivational property that the system uses to yield ordered strings of Lexical Items is *access to the order of application of Merge*, rather than *(asymmetric) c-command relations*, and explore the main consequences of such an approach.

The remainder of this paper is organized as follows: in section 2, I discuss the properties of the architecture of a level-free derivational syntax that I am assuming here. In section 3, I discuss the implications of assuming that it is order of application of Merge, rather than c-command, that yields linear order. In section 4, I formalize the approach I am developing here, and in section 5 I discuss the consequences of this approach for the linearization of Chains, which will also help understand what the differences might be between a representational and a derivational system. Finally, section is devoted to the conclusions and questions for further research.

2. *The architecture of a level-free syntax and departures from it*

2.1. Interpretation of features at the interfaces

Following the main ideas of Epstein et al. (1998), I am assuming here an architecture where there are no levels of representation (i.e., no PF or LF), and so the way that the Sensory-Motor Systems (SMS) and the Systems Of Thought (SOT) interpret the features on Lexical Items is by accessing the information provided by each Merge (or Move) operation. This is one of the fundamental distinctions between this approach and Chomsky's (1995) early minimalism approach; the syntax does not provide the performance systems with a specific level of representation for them to interpret. Under this approach, the performance systems actually "look into" the operations that occur in a derivation and extract from such inspection the information that is relevant to them.

The question that immediately arises is *when* this information is accessed and *what kind of information* present in Merge can be accessed. Epstein et al.'s answer to this question is that *each* operation of Merge provides an instruction to the performance systems SOT and SMS when it applies. Let us refer to this as the *online* approach to interpretation.

[2] Pesetsky and Fox (to appear) discuss different predictions that can be made by examining the output of linearization and the point(s) of the derivation where linearization applies. Their proposal does not commit to a specific approach to the mechanisms that underlie linearization.

We could also imagine an approach where that information is accessed by the performance systems at the end of the derivation. Let us call this hypothesis the *end-of-the-line* approach to interpretation. Notice that this *end-of-the-line* approach is different from the standard weak derivational approach of early minimalism with levels of representation LF and PF (Chomsky 1995) in that the performance systems still access the information provided by *each operation* of Merge, rather than the *structure* created by Merge (recall the quotes from section 1 above).

The table in (1) below illustrates four different possibilities, based on when the performance systems access the derivation (on-line vs. end-of-the-line) and whether there levels of representation or not (derivational vs. representational). Chomsky's (1995) early minimalist approach would be both *representational* and *end-of-the-line*. Conversely, the derivational approach of Epstein et al. (1998) would be *derivational* and *online*.

(1)

	On-line	**End-of-the-line**
Derivational (No PF or LF)	Output of each Merge	Output of each Merge
Representational (PF & LF)	Not Possible	Tree structure created by Merge

2.2.Minimizing Bare Output Conditions

One of the foundational assumptions in the Minimalist Program is that SOT and SMS impose different Bare Output Conditions on the design of the Faculty of Language: SOT impose *hierarchy* and SMS impose *linearity*, and these conditions are understood to shape the way in which the Faculty of Language is designed; the Faculty of Language creates hierarchical structures through Merge, so that SOT can interpret the semantics of a derivation and then linearization applies to that hierarchical structure in order to provide a linear order of terminals that SMS can interpret.

Here I make an even more fundamental assumption about Bare Output Conditions: SOT require features with information about *meaning* and SMS require features with information about *sound*.[3] Accordingly, Lexical Items (and ultimately, linguistic expressions) display Semantic Features and Phonological Features (PhonFs).[4] The hierarchical and linear properties that linguistic expressions display are a consequence of the way the features of Lexical Items are manipulated by the computational system of FL, rather than Bare Output Conditions imposed by the interfaces.

This computational system takes Lexical Items and phrases (i.e., syntactic objects) and puts them together in the simplest way possible, in accordance with minimalist assumptions, by means of the operation Merge. I am assuming Merge cannot apply freely and there are actually two fundamental ways in which Merge is constrained:

[3] Notice that linearity/sound and hierarchy/meaning are *not* necessary correlations under 'virtual conceptual necessity', so we should aim at explaining/deducing them.

[4] Lexical Items also display Formal Features, such as Case Fs. Here I follow Nunes (1999, 2004) in assuming that Case Fs are not interpretable by any of the interfaces, and the syntax has to get rid of them in order for the derivation to be interpreted at the interfaces.

First, Merge is binary; it takes two syntactic objects and puts them together to form a larger unit.[5]

Second, Merge cannot apply freely (see Frampton & Guttman 2002; Collins 2002); in order for Merge to apply, one of the syntactic objects has to select the other. Accordingly, Lexical Items display Selectional Features (SelFs) by means of which one LI requires another LI or phrase to merge with it, and these SelFs are semantic in nature.[6] Verbs, for example, need to merge with a phrase that will be interpreted as its argument, a quantifier may select another phrase (like a PP or a relative clause) to function as its restrictor, etc.

PhonFs, conversely, never drive a Merge operation, which suggests that PhonFs are not relevant when Merge applies.[7]

2.3. Interpreting Merge

Given the above considerations about the nature of Merge, I would like to put forward the following hypothesis:

(2) SOT interpret Merge operations when they apply (*online* approach); SMS interpret Merge operations at the end of the derivation (*end-of-the-line* approach).

If Merge applies in order to satisfy SelFs, which are semantic in nature, as I argued in the previous section, it makes sense to say that a given operation of Merge is interpreted by SOT at that point. Moreover, if Merge applies to two syntactic objects and this operation is interpreted by SOT when it applies, hierarchy in the grammar turns out to be the by-product of successive applications of (binary) Merge interpreted one by one by SOT. Under this view, hierarchy would no longer be a primitive, that is, a Bare Output Condition imposed by SOT.

There is also another reason for preferring this *online* approach to semantic interpretation. One of the features of human language is its constituency structure, which is defined on syntactic trees on the basis of dominance relations. This is clearly a *formal* definition of constituency structure, which cannot be maintained under a derivational approach, due to its representational nature. However, under the derivational approach we can adopt a *substantive* definition of constituency, which is based on the way that Semantic features are incrementally interpreted by the SOT interface.

To illustrate this, consider the beginning of the derivation for a sentence like *I love these movies*:

(3)

Merge Operation	Output of Merge	SOT interpretation
1. (these, movies)⇒	{these, movies}⇒	[these movies]
2. (love, {these, movies})⇒	{love, {these, movies}}⇒	[love [these movies]]

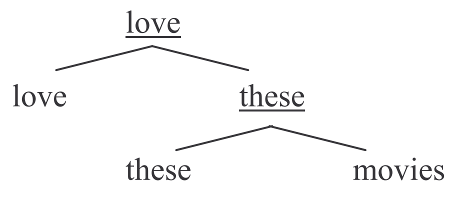
Consider also a representational counterpart of (3). In this case, we would say that the following syntactic tree is created at step 2 above:

[5] I am abstracting away here from whether Merge also provides the output of the operation with a label, based on one of the syntactic objects that are merged (see Collins 2002).

[6] Sometimes Merge is also required by Formal Features, as in the case of functional heads.

[7] At most, PhonFs may require adjunction operations, as in the case of morphological requirements like stranded affixes, but they never drive Merge operations.

(4)



The hierarchical information that the tree in (4) provides is that [these movies] is a constituent and so is [love [these movies]], and also that [love movies] and [love these] are *not* constituents. Notice that in order to know this we need the representation of the syntactic tree and also an axiom that determines that *two or more elements form a constituent iff there is one node α such that those and only those elements are dominated by α.*

Conversely, notice that given (3) above, the same predictions can be made by just considering what SOT *has* and has *not* interpreted. We do not need a syntactic tree, nor do we need an axiom that defines what counts and does not count as a constituent. We could state this derivational, interface-based definition of constituency as in (5) below:

(5) A given set of Lexical Items $\Sigma = \{LI_1, LI_2 \ldots LI_n\}$ forms a constituent iff at some point in a derivation the Semantic features of only the Lexical Items in that set Σ have been interpreted by SOT.

As for SMS, recall that PhonFs never drive a Merge operation, as argued in the previous section. There are other types of operations that are related to sets of PhonFs, like morphological processes, although they are not restricted by constituency structure in the same way that syntactic operations are. All I am claiming here is that two or more Lexical Items (or rather, their PhonFs) do not need to form a constituent at all in order for some morpho-phonological operation to apply to them. Actually, adjacency seems to be a requirement, instead. "*Wanna* contraction", for example, has applied to *want* and *to* in the sentence in (6) below although *want* and *to* were not merged together at any point:

(6) I want to eat a sandwich ⇒ I wanna eat a sandwich
Merge Operations: Merge$_1$ (a, sandwich), Merge$_2$ (eat, {a, sandwich}), Merge$_3$ (to, {eat, {a, sandwich}}), Merge$_4$ (want, {to, {eat, {a, sandwich}}}) …
(notice: no Merge$_n$ (want, to))

I am assuming here that, unlike SOT, SMS interprets the PhonFs of Lexical Items at the end of the derivation (recall the end-of-the-line approach). Notice how this relates to the central assumption in the Distributed Morphology framework (Halle & Marantz 1993) that PhonFs are inserted at the end of the derivation (i.e., late insertion). It also resembles the architecture proposed by Groat and O'Neil (1996) in which Spell-Out applies at the LF interface, that is, at the end of the derivation.

An important question that arises under this level-free approach is the status of Full Interpretation (FI), as a condition on the interfaces.[8] To the extent that the hypothesis about SOT-related information above is on the right track, FI seems to be restricted to SMS. If indeed SOT interprets the derivation as it proceeds, then some uninterpretable Feature on an LI, say Case-F on a DP, would make the output of the operation of Merge not interpretable.

[8] Notice that FI cannot be formulated as a condition on LF or PF representations, under this level-free approach.

Under this strong derivational approach to syntactic relations, then, it does not make sense to assume that FI holds for SOT. Conversely, it is widely assumed (see Chomsky 1995, 2000; Uriagereka 1998; Nunes 1999, 2004 and others) that dislocation is related to uninterpretable Fs (Case Fs on DPs for instance). If this is true, then FI seems to be relevant for SMS in the sense that only Lexical Items with just PhonFs can be interpreted. I return to the important relation between SMS and FI in section 5.

3. Linear order derived from order of Merge

Given the architectural assumptions made in the previous sections, I would like to propose the following linearization algorithm:

(7) SMS interpret (the PhonFs of) Lexical Items in the opposite order in which Merge operations inserted them in the derivation.

Assuming (7), we can say that the linear order that a given expression displays is directly related to the order in which Lexical Items entered the derivation and not to the structural relations that hold among them. A welcome result of this approach is that it eliminates all the computation that applying Kayne's LCA to a syntactic tree involves. This is illustrated more clearly in section 5 below, when I discuss linearization of Chains.

An important question that arises under this strong derivational approach is: how is the information about order of Merge retained (Acrisio Pires, p.c.)? I consider three possibilities here:

(a) We can assume that the syntax is purely derivational and there is no representation of syntactic structure at all. If this is the case we would need the order of Merge operations, i.e., the information about what LI merged when, to be retained by some other mechanism. Under this view, there would be a separate component of the grammar that would have stack-like properties, Lexical Items would be pushed into a stack and then they would be interpreted "backwards" by SMS. While positing such a component would require justification, a very attractive property of this approach is that it would explain why Lexical Items are pronounced in the opposite order in which they enter the derivation. That would be the way in which Linearization works simply because the procedure has the standard and expected computational properties of stack-based algorithms in general.

(b) We can weaken the derivational approach and say that there actually is representation of syntactic structure, and the syntax makes use of it, but the interfaces can only interpret the output of Merge operations. In this case we would be making the non-trivial claim that the conditions regulating syntactic operations (say, c-command, economy, relativized minimality, etc.) and the conditions regulating interpretation of the features of Lexical Items by the interfaces are completely different in nature; the former would be defined based on representation of syntactic structure, while the latter would take into account basic properties of Merge.

(c) A final possibility, which relates to current assumptions in minimalism, is that there is representation of syntactic structure only up to the phase level, i.e., until *v** or Comp project (see Chomsky 2000, 2001), and then that representation is no longer available.

It is not necessary to commit to any of these approaches for the scope of this paper, given that they all share the feature that the performance systems can only interpret the output of Merge, which is the crucial feature for my approach to work. Exploring the exact conceptual and empirical consequences of each of the above approaches is a task that I leave for further research.

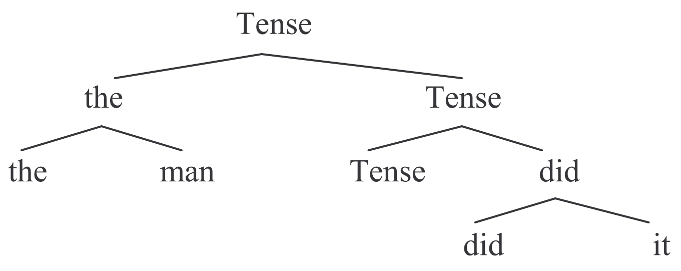
4. C-command vs. order of Merge

It is been assumed since at least Chomsky (1995) that linear order is achieved through linearization of a hierarchical structure (with no linear order information) created by Merge. As pointed out in the introduction, most approaches to linearization rely on (asymmetric) c-command in order to yield a linear ordering (see Chomsky 1995; Guimarães 1998; Nunes 1999, 2004; Richards 2001 and others). A more derivational version of this approach is Epstein et al. (1998), where it is argued that c-command is not a primitive, but it can be derived from order of Merge. There have been two main "traditional" complications with the idea of using (asymmetric) c-command for linearization purposes.

The first one is the "symmetry at the right edge" problem. Under Bare Phrase Structure, when the first two Lexical Items merge in a derivation, there is no asymmetric c-command relation between them, so they cannot be linearized. I am not going to try to solve this problem here (see Chomsky 1995; Guimarães 1998; Epstein et al. 1998, and Moro 2000 for different solutions).

The second one is the "branching specifiers" problem. In order to illustrate this problem, take a simple sentence like (8) and its derivation (ignoring now subject movement for simplicity):

(8) the man did it



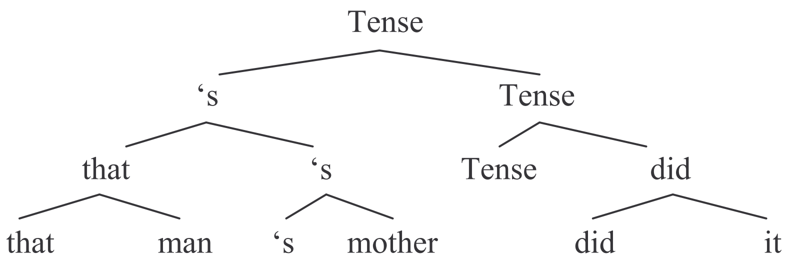
In order to build this syntactic tree, we need to assemble the syntactic object {the, man} and the syntactic object {Tense, {did, it}} in parallel workspaces, in order to get the correct constituency structure (see Uriagereka 1998). As can be seen, when there are branching specifiers, we have a situation in which some LI (say, *the*) can precede another LI (say, *did*) without c-commanding it. This is an obvious complication for any LCA-based approach to Linearization, and there have been several attempts to solve this problem.

Kayne (1994) proposes that specifiers are actually inserted in the derivation by adjunction. The result is that the adjoinee c-commands the element it adjoins to, given the segment/category distinction (see May 1985). This will allow the terminals inside the specifier to precede the rest of the structure.

Uriagereka (1999) proposes that branching specifiers are spelled out independently, hence the name of its approach: Multiple Spell-Out. Uriagereka assumes that there is a one-to-one correspondence between number of workspaces and number of applications of Spell-Out; if a syntactic object is built in a different workspace, it will automatically trigger a separate application of Spell-Out.

Here I also assume, like Uriagereka does, that specifiers are built in different workspaces. However, I argue that they are inserted in the derivation after heads and complements are assembled. The result is that specifiers are *always* inserted in the derivation after heads and complements are *regardless of their phrasal status*. The main piece of evidence for claiming that Merge operations in a workspace are ordered with respect to another actually comes from instances of 'specifers inside specifiers'. Consider in this respect the sentence in (9) and its derivation:

(9) that man's mother did it



In this case, there have been three objects assembled independently in three workspaces: {that, man}, {'s, mother} and {Tense, {did, it}}. Notice that in order to get the correct constituency structure, the objects {that, man} and {'s, mother} obviously have to merge *before* the resulting object merges with {Tense, {did, it}}. This suggests that there is logical ordering between Merge operations in different workspaces (see Fernández-Salgueiro 2004 for more details regarding this proposal).

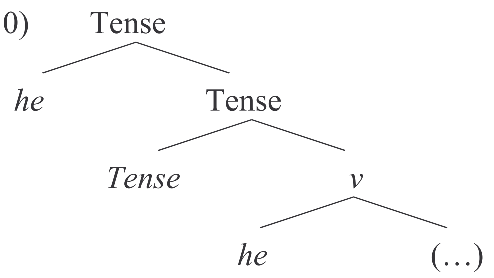
Under this approach, the problem pertaining to terminal elements inside specifiers that I pointed out above does not arise. In the tree in (8) above, for example, both *the* and *man* were inserted in the derivation after *did* and *it* were, so a linear order can be established by the algorithm in (7). In other words, under my approach there is indeed a relation (i.e., order of Merge) between the Lexical Items inside a branching specifier and the Lexical Items inside, say, a complement. This was not present in Kayne's (1994) or Uriagereka's (1999) proposals, which led to different complications for the linearization procedure.

It is important to point out, however, that I am not claiming that c-command should be eliminated from the system at all. If there is one fundamental relation that regulates syntactic operations, that is c-command, and the proposal here does not make any claim about the nature and/or conditions regulating such relations. In my proposal I seek to eliminate only *asymmetric* c-command, by arguing that it is not the relation that the system uses to yield linear order from hierarchical structure. As far as I know, *asymmetric* c-command has not been argued to play any other role in the system.

5. Phonetic realization of Chains

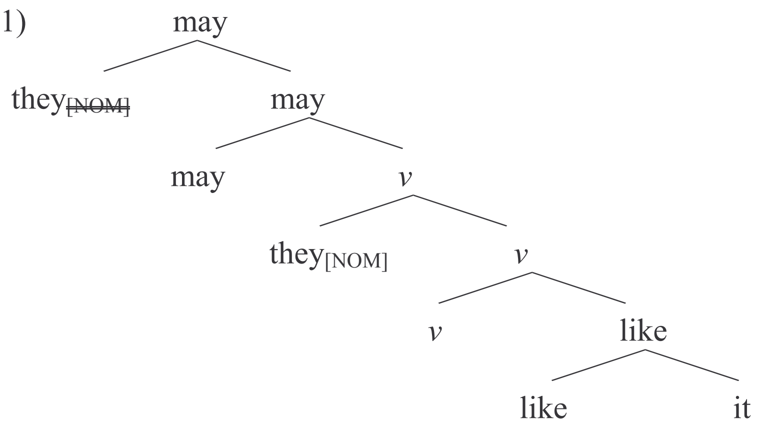
In this section I would like to discuss the implications of my approach for a theory of Chain linearization, and whether it makes any predictions which of the member(s) of a Chain is (are) interpreted by SMS. I am going to take here Nunes's (1999, 2004) theory of linearization of Chains as a starting point.

Nunes's proposal crucially relies on the fact that *all* c-command relations among Lexical Items are computed in order to yield an ordered sequence of terminals. In order to illustrate this, consider the tree in (10):

(10)


When the LCA applies to this tree structure, it is unable to decide whether *he* precedes or follows *Tense*, since there is a c-command relation contradiction: *he* c-commands Tense but *he* is also c-commanded by *Tense* at the same time. In Kayne's terms, there two sets of ordered pairs that involve a contradiction: <he,Tense> and <Tense,he>. Nunes argues the LCA cannot assign a linear ordering unless one of the copies of *he* is deleted and hence not interpreted at the interface. Notice that for this contradiction (and eventual deletion of the lower copy of *he*) to occur, *all* c-command relation among (the upper) *he*, *Tense* and (the lower) *he* have to be computed. As I show here, my approach makes the same empirical prediction without necessitating the computation of all asymmetric c-command relations.

Let us consider a more concrete example. Consider the derivation for *they may like it*:

(11)


Before we continue, it is important to note that Nunes is assuming a Checking-based (not an Agree-based) approach to the deletion of uninterpretable Fs like Case Fs, and no Chain Uniformity Condition. Under his approach, only the upper copy of *they* has had its Case Fs deleted in (11). In my previous work (Fernández-Salgueiro 2005) I have argued that, even under an Agree-based approach, the lower copy of *they* does not get its Case Fs deleted; only the upper one does. For other approaches that claim Case checking to occur in a Spec-Head configuration see Rezac (2003); Müller (2004) and Epstein & Seely (2006).

Nunes's question is: why is the linearization output of (11) *they may like it* and not **they may they like it*, **may they like it* or **may like it*? Nunes's approach is gong to provide a principled answer to this question based on economy considerations and (un)interpretability of Fs at the SMS interface. He proposes that the minimum amount of deletion operations should be employed provided the result satisfies FI at PF.[9]

The first ungrammatical option, **they may they like it*, is ruled out independently by the LCA, as we saw at the beginning of this section (recall the discussion on example (10)). If we chose **may they like it*, we need two deletion operations, one which deletes the upper copy of *they* and another one which deletes the NOM-F of the lower copy of *they* (which has not been deleted in the course of the derivation). If **may like it* is chosen, we also need two operations, one which deletes the upper copy of *they* and another one which deletes the lower copy of *they*, together with its NOM-F. If we choose *they may like it*, though, only one deletion operation is necessary to make the structure linearizable, the one which deletes the lower copy of *they*. In sum, *they may like it* is the option with fewer deletion operations that satisfy FI at PF, i.e., it yields a PF representation with no uninterpretable Fs.[10]

As can be seen, Nunes provides an elegant explanation of why top-most copies are the ones that are generally pronounced. There are, however, a couple of conceptual problems with this approach. First, it is not clear how to calculate the number of deletion operations. What counts as *one* deletion operation? A syntactic node? A feature? A feature bundle?[11]

Second, the LCA seems to be unable to detect "differences" between the Lexical Items in their feature make-up (e.g., whether a Case F has been deleted or not). Then, when an LCA contradiction is encountered, Nunes allows "analysis" of the feature make-up of the Lexical Items that cause that linear order contradiction. In other words, when Linearization first applies, only PhonFs are visible. However, after the contradiction is detected, the whole feature make-up is visible and so are features like Case. This involves an architectural paradox, with features becoming invisible and then visible again as the derivation to PF proceeds.

I would like to rethink Nunes's proposal in more derivational terms, by also relying on the (un)interpretability of Fs. I will show how these two problems do not arise under my derivational version of Nunes's theory. I will start with an assumption that follows from the principle of FI: an LI cannot be interpreted by SMS if it contains uninterpretable material, that is, not just PhonFs.[12] I will try to show that the properties of the linearization of Chains will follow from these FI-based considerations alone.

Following the algorithm in (7) discussed in section 3 above, SMS are going to interpret the PhonFs of Lexical Items in the reversed order in which they were merged. Accordingly, for the derivation in (11) above, repeated here as (12), the upper copy of *they* would be the first LI whose PhonFs are interpreted by SMS. As its NOM-F has been deleted, it is fully interpretable, and no problem arises here. Then the PhonFs of *may* are also interpreted by SMS with no problem, since its uninterpretable phi-Fs have been deleted. However, when SMS try to interpret the lower copy of *they*, an uninterpretable NOM-F is found. Given FI, the lower copy of *they* cannot be interpreted by SMS.

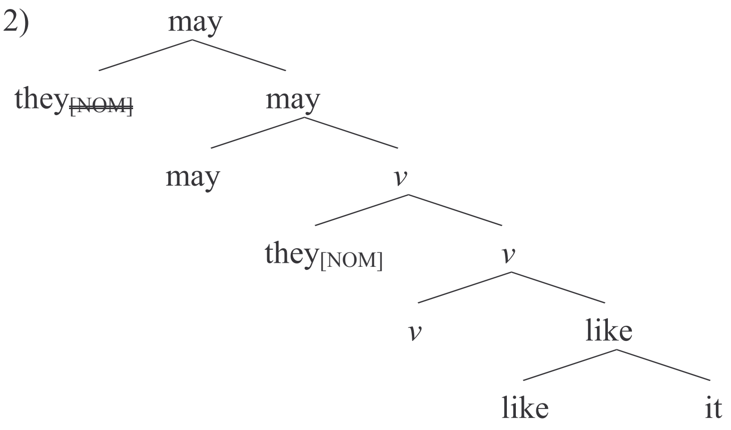
[9] Nunes assumes a weak derivational approach with levels of representation PF and LF.

[10] We can also think about this condition in terms of valued vs. unvalued features (Chomsky 2001). Lexical Items with valued formal features are interpretable by SMS, Lexical Items with unvalued formal features are not.

[11] I owe this critique to Sara Rosen (pc).

[12] Recall that in my approach FI does not apply to SOT. Semantic features are interpreted by SOT when they are merged, regardless of any UnFs an LI may display.

(12)



Notice that, under this approach, the upper copy of the chain is pronounced independently of the status of the other Lexical Items (and therefore, independenty of the status of its original copy), so no further computation is needed to determine whether or when the upper copy is interpreted. Under this approach, SMS try to interpret each LI, independently of whether they were inserted by Merge or Move. In this respect, it is interesting to note that Move (internal Merge) has been usually defined as (at least some version of) a COPY+MERGE mechanism. The result is that (order of) Merge operations, regardless of whether they insert a copy of an LI or a new LI, provide SMS with PhonF information in the same way.

Moreover, the architectural paradox explained above does not arise. The whole feature make-up of an LI is visible when linearization applies. Under this approach, then, FI is a condition on *each* Merge operation; if an LI that contains UnFs is associated with a given operation of Merge, all that FI does is prevent that LI from being interpreted by SMS.

Besides the technical differences, then, both Nunes's approach and my approach predict that *all and only* Lexical Items without uninterpretable features are pronounced, and there is no reference whatsoever to which copy is the highest or the lowest, and so no stipulation that only the upper copy of a chain can be pronounced (as was the case with *trace* theory in GB). The main difference between Nunes's and my approach is that I am proposing that the decision of which copy to pronounce depends solely on interface conditions, while Nunes claims that economy of deletion operations also plays a role. To the extent that the problems with Nunes's proposal can be overcome, my approach to linearization seems to be more optimal.

6. Conclusions

In this paper I have developed a version of the level-free derivational approach to syntax, in which there is an asymmetry as to when Semantic features and PhonFs are interpreted by SOT and SMS, respectively. I have also explored some consequences of such an approach for the way we understand the architecture of FL.

I have also developed a derivational linearization procedure under this approach, one in which it is the logical order of Merge operations, rather than (asymmetric) c-command relations, that provides a sequence of terminals for a given derivation. It is important to notice that the approach developed here understands Linearization not as a rule, but rather, as an interpretive procedure. This approach should therefore be understood as an answer to the following question: how does SMS come up with a linear order of Lexical Items when it inspects and looks into a derivation? I have shown that this derivational approach has rich empirical consequences for the linearization of Chains.

Acknowledgements

I would like to thank Cedric Boeckx, Sam Epstein, Michael Marlo, Jairo Nunes, Acrisio Pires, Sara Rosen, Daniel Seely, and Juan Uriagereka for helpful comments on previous versions of this paper. I would also like to thank the audience at ConSOLE XIV for their suggestions. All errors are mine.

Gerardo Fernández-Salgueiro
University of Michigan
gfthrak@umich.edu
http://www.umich.edu/~gfthrak

References

Chomsky, N. (1995). *The Minimalist Program*. MIT Press, Cambridge, MA.

Chomsky, N. (2000). Minimalist Inquiries: the framework. R. Martin, D. Michaels & J. Uriagereka (eds.), *Step by step: essays on minimalist syntax in honor of Howard Lasnik*. MIT Press, Cambridge, MA, pp. 99-156.

Chomsky, N. (2001). Derivation by phase. M. Kenstowicz (ed.), *Ken Hale. A life in language*. MIT Press, Cambridge, MA, pp. 1-52.

Collins, C. (2002). Eliminating labels. S. Epstein & T. D. Seely (eds.), *Derivation and explanation in the Minimalist Program*. Blackwell, Oxford, pp. 42-64.

Epstein, S., E. Groat, R. Kawashima & H. Kitahara (1998). *A derivational approach to syntactic relations*. Oxford University Press, Oxford.

Epstein, S. & T. D. Seely. (2002). Rule applications as cycles in a level-free syntax. S. Epstein & T. D. Seely (eds.), *Derivation and explanation in the Minimalist Program*. Blackwell, Oxford, pp. 65-89.

Epstein, S. & T. D. Seely (2006). *Derivations in minimalism*. Cambridge University Press, Cambridge, UK.

Fernandez-Salgueiro, G. (2004). What should I merge where? Ms. University of Michigan.

Fernández-Salgueiro, G. (2005). Agree, the EPP-F and Further-raising in Spanish. R. Gess & E. Rubin, (eds.), *Experimental and theoretical approaches to Romance linguistics*. John Benjamins, Philadelphia, pp. 97-107.

Frampton, J. & S. Guttman (2002). Crash-proof syntax. S. Epstein & T.D. Seely (eds.), *Derivation and explanation in the Minimalist Program*. Blackwell, Oxford, pp. 90-105.

Guimarães, M. (1998) *Repensando a interface sintaxe-fonologia a partir do Axioma de Correspondência Linear*. MA Thesis. Universidade Estadual de Campinas.

Groat, E. & J. O'Neil (1996). Spell-Out at the LF interface. W. Abraham, S. Epstein, H. Thráinsson & C. J-W. Zwart (eds.), *Minimal ideas: syntactic studies in the minimalist framework*. John Benjamins, Amsterdam, pp. 113-139

Kayne, R. (1994). *The antisymmetry of syntax*. MIT Press, Cambridge, MA.

May, R. (1985). *Logical form: its structure and derivation*. MIT Press, Cambridge, MA.

Moro, A. (2000). *Dynamic antisymmetry*. MIT Press, Cambridge, MA.

Nunes, J. (1999). Linearization of chains and phonetic realization of chain links. S. Epstein & N. Hornstein (eds.), *Working Minimalism*, MIT Press, Cambridge, MA, pp. 217-250.

Nunes, J. (2004). *Linearization of chains and sideward movement*. MIT Press, Cambridge, MA.

Pesetsky, D. & D. Fox (To appear). Cyclic linearization of syntactic structure. K. É. Kiss, (ed.). *Theoretical Linguistics*, special issue on Object Shift in Scandinavian.

Richards, N. (2001). A distinctness condition on linearization. Paper presented at GLOW 24, Universidade do Minho, Braga, Portugal. http://web.mit.edu/norvin/www/papers/Distinctness.pdf

Uriagereka, J. (1998). *Rhyme and reason*. MIT Press, Cambridge, MA.

Uriagereka, J. (1999). Multiple Spell-Out. S. Epstein & N. Hornstein (eds.), *Working Minimalism*. MIT Press, Cambridge, MA, pp. 251-282.