Serially bottom-up:

Generating Lexical Arrays and restricting Derivational Workspaces

Gerardo Fernández-Salgueiro

Abstract

In this paper I subject the notions lexical array, Merge, and derivational workspace to minimalist scrutiny. I argue that both lexical arrays and Merge are constrained by selectional features and that derivations start by targeting only one lexical item in a single workspace, which will initiate only the exact number of workspaces needed for a derivation to converge. Combined with a linearization algorithm based solely on order of Merge, this approach provides a novel solution to problems for previous linearization approaches, and derives specifier-head-complement order regardless of phrasal status. Finally, I discuss some implications for approaches to spell-out and extraction domains.

Keywords: Merge, lexical array, derivational workspace, linearization, LCA, sub-extraction.

1. Introduction

Minimalism makes extensive use of the notions of lexical array (LA), Merge and derivational workspaces (DWSs). Of these three, only the mechanics of Merge have been investigated in detail, to the best of my knowledge. In this paper I study the properties of Merge, LAs, and DWSs and subject them to standard minimalist scrutiny.

Under current Minimalist assumptions, the computational system can generate expressions that concatenate arbitrary combinations of lexical items in an arbitrary number of DWSs, which I show leads to computational inefficiency. I argue instead that

(i) Merge only applies in order to satisfy selectional features (SelF) of lexical items and (ii) derivations start by targeting only one lexical item in a single DWS. I also argue that this approach is computationally more efficient since it only initiates the exact number of DWSs that a derivation needs to converge given the initial LA. Finally, I propose a linearization algorithm purely based on order of Merge, which will overcome two of the most pervasive problems for linearization: symmetry at the right edge and phrasal specifiers.

2. The Empirical Motivation for DWSs

It is easy to show that the system would massively undergenerate if it didn't make use of different DWSs. Consider the derivation in (1), for example:[1]

(1) {love, Mary}
{*v*, {love, Mary}}
{I, {*v*, {love, Mary}}}

In this sequence of operations, we have just been adding lexical items to the same DWS. We thus have what Uriagereka (1999) calls a command unit, with continuous application of Merge in a single DWS.

Now suppose that the external argument of *v*, instead of being just one lexical item, is phrasal, something like *these men*, and we try to continue in the same DWS, with Merge of *men*:

(2) {love, Mary}

{*v*, {love, Mary}}

{men, {*v*, {love, Mary}}}

This option is obviously not going to work. If we now add the lexical item *these* to the DWS, we would get the syntactic object {these, {men, {v, {love, Mary}}}} as output of Merge, which obviously doesn't represent the correct constituent structure for the *v*P *these men love Mary*.[2]

In order to insert the phrasal element *these men* in the derivation, we need to create the syntactic object {these, men} separately and then merge it in the original command unit, as in (3):

(3) DWS_1 DWS_2

{love, Mary} {these, men}

{*v*, {love, Mary}}

{{these, men}, {*v*, {love, Mary}}}

The standard assumption in Minimalism is that the two DWSs are created in parallel, and that there is no order relation between the operations of Merge that apply in different DWSs. One could say then that in standard Minimalism derivations are bottom-up but also horizontal. In this respect, notice that the object {these, men} in DWS_2 was created

independently of the properties of DWS_1. In section 5 I show that this property leads to computational inefficiency.

Before we start the discussion of DWSs proper, I would like to examine the properties of Merge (section 3) and LAs (section 4).

3. Questions about the nature of Merge

In standard Minimalism, Merge is understood to be free, unlike Move. This means that the system relies heavily on the role of the interfaces as filters, something that has been argued against in the literature. Perhaps the most explicit critique of this approach is that of Frampton and Gutmann's (2002), who claim that SelFs must be satisfied by means of Merge. I agree with Frampton and Gutmann that if Merge is indeed constrained by SelFs, the system becomes more computationally efficient, at least, within one DWS.[3]

Following these ideas, I will be assuming that the operation of Merge is not free, but actually constrained by SelFs, making Merge inherently asymmetric. A version of this approach is actually what Chomsky proposed in early Minimalism, as can be understood from the following:

> "The operation Merge(α, β) is asymmetric, projecting either α or β, the head of the object that projects becoming the label of the complex formed. If α projects, we can refer to it as the target of the operation…" (Chomsky (1995:246)

Another crucial assumption about Merge is that it is a binary operation. While I believe there is little doubt by now that complex linguistic expressions are composed of binary

branching structures, it's true that this doesn't necessarily entail that Merge itself is binary. In this respect, a number of researchers, like Frampton and Gutmann (2002) and Zwart (2004, 2011), have proposed the existence of Unary Merge, which is responsible for inserting lexical items one by one in a derivation.[4]

Unary Merge can thus create a DWS without concatenating two lexical items, which is taken as a welcome result for a number of reasons. Here I comment on two of those. First, derivations can then be understood to proceed by introducing heads that select for previously introduced elements, which ensures that selectional requirements will be efficiently met since "heads are introduced automatically in the right order." (Frampton and Gutmann 2002:97). Second, Unary Merge derives the asymmetry that the syntax needs for, among other things, computing linear order at the PF interface, and solves the "symmetry at the right edge" problem, which is one of the most pervasive problems for linearization (see section 6 below).

Another reason why in principle we might think that we need some sort of Unary Merge is the existence of linguistic expressions that don't have internal syntactic structure, such as *sorry*, *damn*, *hey*, *hi*, *oops*, *sure*, *right* (while nodding), *what?*, *yes*, *no*, etc. These expressions, though not syntactically derived, do undergo spell-out and receive a PF and LF interpretation. We could even argue that Unary Merge is more basic than standard Merge, since not all linguistic expressions are composed of two or more lexical items. In order to understand these versions of Merge better and arrive at a possible unified approach, we need to examine Lexical Arrays (LAs), which are the starting point of any derivation, and how they relate to DWSs.

4. From the Lexicon to the Lexical Array to the DWS: motivating Merge

In standard Minimalism, lexical items are understood to first form a LA, which constitutes the Reference Set of a derivation. Lexical items are then selected from the LA and undergo syntactic operations. The generation of this LA is totally unconstrained, the reason being that if we imposed any well-formedness conditions on LAs this would mean going back to a somewhat camouflaged version of Deep Structure.

I agree that in a derivational theory of syntax the syntactic properties of linguistic expressions should emerge from the way they are manipulated in the derivation and thus LAs should be devoid of any syntactic conditions (such as the X' template or the θ-criterion in GB). However, a system that does not impose any restrictions whatsoever on the Reference Set of a derivation is anything but computationally efficient, I believe, considering the amount of lexical items in the lexicon and all the possible combinations.

How is one to solve this tension then between entertaining a purely derivational theory (with no Deep Structure) and establishing conditions on LAs that can yield a more efficient system? I suggest that a way out of this tension is to use information that is already present in the lexical items themselves, that is, their SelFs.[5] If we examine the LAs that constitute the Reference Sets of converging derivations we find that each one of its lexical items either selects or is selected by another lexical item.

Let's now look at an example that will help us refine this statement. Consider the three LAs in (4), where subscripts indicate the category of the lexical item and brackets indicate the category they select for (if any):[6]

(4) a. LA = {$_{D}$he, $_{T}$will[v], $_{v}$v[V][D], $_{V}$love[D], $_{D}$a[N], $_{N}$woman}

b. LA = {$_{D}$he, $_{N}$tie, $_{T}$will[v], $_{v}$v[V][D], $_{V}$love[D], $_{D}$a[N], $_{N}$woman}

c. LA = {$_{D}$he, $_{T}$will[v], $_{v}$v[V][D], $_{D}$a[N], $_{N}$woman}

In (4a), each category in brackets can find a subscript counterpart in another member of the set. The opposite is also true for every member except for one of them, the T head *will*. In (4b), each category in brackets can also find a subscript counterpart in another member of the set; however, there are now two subscripts that can't find a bracketed counterpart (*will* and *tie*). In (4c), the opposite problem arises: *v* can't find a V subscript. Interestingly, (4b) and (4c) won't yield a convergent derivation, while (4a) will, the result of which being a TP headed by *will* (the only element that is not selected for). We can then say that *only LAs in which each SelF is mirrored by one and only one lexical item are allowed*. Again, I see no reason why the computational system (even a strongly derivational one) shouldn't make use of this information already present in lexical entries in order to construct LAs more efficiently. In this paper I will assume that it does.

Notice that I haven't said anything about possible "size requirements" of LAs. In this respect, it has been proposed that LAs are actually composed of subarrays (see e.g. Chomsky 2001). If this is correct, it means that we need to allow for one (and only one) of the SelFs present in a subarray to correspond to the label of the object that has resulted from exhausting a previous subarray. For example, a T head requiring a *v* is allowed in a (C-related) subarray because the syntactic object already formed is a *v*P.[7]

The next step in a derivation is to build a syntactic structure out of the LA. Again, we have different theoretical options; we could claim that any lexical item can be targeted for Merge or we can assume that only certain lexical items can initiate a derivation. As already pointed out, standard Minimalism imposes no restrictions. Frampton and Gutmann (2002), however, claim that only a lexical item that doesn't display SelFs can start a derivation.

If we follow the latter option, we must assume that there are two different kinds of Merge: the kind that initiates a DWS (inserting a lexical item by itself in a derivation) and the kind that expands an already created DWS. This could be taken to be an unwelcome result conceptually, since it would mean that we have two structure building operations: (Unary) Merge(α) and Merge(α, β). Moreover, the former operation wouldn't satisfy any SelFs, while the latter would (actually, in Frampton and Gutmann's system, it *must*).

A way to unify these two kinds of Merge is to make use of the notion Target α, though not exactly the way Chomsky (1995) defines it (see section 3).[8] For Chomsky, Merge is asymmetric in the sense that its result (a projecting structure) is. Let's assume a stronger version of asymmetric Merge and claim that Merge applies in order to satisfy the SelFs of the element that has been targeted. To illustrate, consider how the DP *the men* would be derived:

(5) LA = {… $_{D}$the[N], $_{N}$men}

Target($_{D}$the[N]) $\Rightarrow$ Merge($_{D}$the[N], $_{N}$men) $\Rightarrow$ {$_{D}$the, $_{N}$men}

Is this very same formalism applicable to Unary Merge? I would like to argue here that, despite appearances, Unary Merge is also *binary*. A derivation is the mapping from an LA (a set of lexical items) to a DWS. The DWS starts as the empty set Ø and is gradually expanded, while the LA is gradually reduced to Ø. When we examine converging derivations, we see that lexical items targeted in the LA are merged with the set already assembled in the DWS (see section 3). Again, before any Merge operations take place, a DWS is the empty set Ø. Since we're assuming that Merge satisfies the SelFs of the lexical item that is targeted, Target α when α has no SelFs allows Merge(α, Ø), where the output of Merge is {α}. Under these assumptions, Unary Merge is nothing more than binary Merge of a lexical item with the empty set. Interestingly, Fortuny (2008) reaches a similar conclusion on slightly different grounds (Fortuny doesn't take into account SelFs). This is a welcome result, since we can now keep the advantages of Unary Merge pointed out in section 3 while maintaining the idea that Merge is an inherently binary operation, as Chomsky originally proposed.

It is not clear, though, that this alone entails that only a lexical item with no SelFs can be the first element to enter a derivation (Frampton and Gutmann's claim).[9] To illustrate, consider (6), which is similar to the one used by Frampton and Gutmann.

(6) LA = {$_{D}$they, $_{T}$will[V], $_{V}$arrive[D]}

An LA consisting of three lexical items obviously gives us three logical possibilities. Suppose Target($_{T}$will[V]) applies first. At the next step, the derivation is cancelled since

no lexical item left in the LA selects for T. As we have just seen, Target($_D$they) applying first is a possibility, since it can be merged with the empty set Ø (and then *arrive* and *will* would be merged). Now, if Target($_V$arrive[D]) applies first, Merge($_V$arrive[D], {$_D$they}) would be licensed, and then *will* would be inserted. Unaccusatives, then, seem to provide no clear evidence that only lexical items with no SelFs can initiate a DWS.

The derivation of unergative sentences, however, suggests that a lexical item with no SelFs should the first one to be targeted for Merge. Suppose an LA includes an unergative predicate, like (7):[10]

(7) LA = {$_D$he, $_T$has[v], $_v$v[V][D], $_V$resigned}

Which lexical item should be targeted first, *v* or *resigned*? If Target($_V$resigned) applies first, then the order of subsequent Merge operations is unambiguous. Target α applies to the lexical item that contains a SelF that matches the category of the root element in the DWS, except when the DWS includes a lexical item that has a SelF that still needs to be satisfied, in which case that element is targeted again for Merge.[11] This is illustrated in (8):

(8) Target($_V$resigned) ⇒ DWS = {$_V$**resigned**}

Target($_v$v**[V]**[D]) ⇒ DWS = {**($_v$v[D])**, {$_V$resigned}}

Target($_v$v[D]) ⇒ DWS = {$_D$he, {$_v$**v**, {$_V$resigned}}}

Target($_T$has**[v]**) ⇒ DWS = {$_T$has, {$_D$he, {$_v$v, {$_V$resigned}}}}

Now suppose that Target($_{v}$v[V][D]) applies first. In this case we would have to satisfy two different SelFs, which means there's no unambiguous way for the derivation to proceed. Actually, one of the possibilities would be to have Merge($_{v}$v[V][D], $_{D}$he) applying first, which derives an incorrect order of operations (in the sentence *he has resigned*, the term {$_{v}$v[V], $_{D}$he} is not a constituent). Again, if the derivation targets a lexical item with no SelFs first we can derive subsequent steps unambiguously, with the element already present in the DWS determining which lexical item must be targeted next. As will be seen in section 6, this will be instrumental in deriving the fact that specifiers are built after heads and complements regardless of phrasal status.

An interesting property of derivations that we can deduce is that there is really no such thing as Merge of two lexical items. A DWS is initiated by Merge(lexical item, Ø) and is continued by Merge of a lexical item with whatever set has been formed in the DWS at that point. A welcome result is that the asymmetric properties of syntax (cf. Kayne 1994, Zwart 2011) that were hard to capture under Bare Phrase Structure (partly because Merge of two lexical items was allowed) can now be easily captured under the approach that I am following here (see section 6 for further discussion).

A final advantage of this approach is that all derivations, even one-word ones, are initiated in exactly the same way, by merging a lexical item with no SelFs with the empty set Ø. Moreover, this would also mean that one-word linguistic expressions are also sets formed by Merge, though containing only one lexical item ({right, Ø} ={right}).

Now that we have a restrictive theory of Merge and LAs, let's discuss how different DWSs arise in a derivation.

5. A Minimalist Theory of DWSs

As pointed out in section 2, a syntactic theory in which structure is built bottom-up by means of Merge must allow for more than one DWS to be created; otherwise we would be predicting that phrasal specifiers don't exist, contrary to fact.

The question that immediately arises then is the following: are there any restrictions regarding how many and/or when DWSs can be initiated? In standard Minimalism, there're no restrictions at all. Since the operation of Merge is understood to "come free", nothing prevents it from concatenating an arbitrary number of lexical items in an arbitrary number of DWSs (the horizontal property I alluded to in section 2). Let's refer to this approach as the *parallel* DWS theory. In the other approaches I mentioned in sections 3 and 4, we have a similar situation; nothing prevents multiple instances of "first Merge" in multiple DWSs.[12] In this section I propose a more restrictive system in which the process of initiating additional DWSs is heavily constrained.

5.1. Economy of DWSs

First of all, I would like to propose that the number of DWSs that can be created is subject to economy considerations, thus disallowing their random proliferation. This seems conceptually desirable, since presumably having parallel DWSs increases computational complexity.

Consider a derivation like (9):[13]

(9) LA = {… the, men, believed, their, dad, want, that, boy, expect, his, mom, mad, at, him}

DWS_1 = {men}, DWS_2 = {dad}, DWS_3 = {boy}, DWS_4 = {mom}, DWS_5 = {him}

Further applications of Merge and other syntactic operations would yield an expression like *the men believed their dad to want that boy to expect his mom to be mad at him*. This derivation, with five parallel DWSs would be possible under the approaches I considered in the previous two sections, including the one I have proposed (since all those lexical items could be merged with the empty set Ø in five different DWSs). This exact derivation wouldn't be possible in standard Minimalism because Unary Merge isn't allowed, but the same number of parallel DWSs could be created (DWS_1 = {the, men}, DWS_2 = {their, dad}, DWS_3 = {that, boy}, DWS_4 = {his, mom}, DWS_5 = {at, him})

Now imagine a similar sentence with simple specifiers (pronouns) instead of complex ones, like *they believed him to want him to expect her to be mad at him*, and consider how its derivation would start:

(10) LA = {they, believed, him, to, want, him, to, expect, her, to, be, mad, at, him}

DWS_1 = …{mad, {at, {him}}

In this case, only one DWS is initiated. Interestingly, although (9) and (10) display two radically different derivations, the end result after exhausting their LAs is not really that different (besides the obvious simple vs. complex specifier distinction). Moreover, it

seems out of proportion to make the beginning of the derivation five times more complex just because the final result will be an expression containing four complex specifiers. In other words, in the complex specifier expression complexity is scattered along four different parts of the expression, but its derivation *crams all this complexity in its initial step*.

While I know of no empirical evidence against a system that allows a derivation with an arbitrary number of *parallel* DWSs like (9), I believe that a system that restricts the number of DWSs would be more economical, hence preferable on minimalist grounds. It is then possible to imagine a theory under which (9), in which only one DWS is initiated, can also be the beginning of the derivation for *the men thought that John's dad believed that that boy made his mom mad at him*, with additional DWSs being created as the derivation proceeds, and only when a complex specifier is created. In more computational terms, we could say that creating an additional DWS is a *subroutine* triggered by the need to satisfy a SelF that is inside the current DWS. This way, the system only has to consider two DWSs at most simultaneously (for this example), thereby reducing computational complexity.[14] Let's dub this approach, in which derivations proceed in a bottom-up fashion from a single DWS, the *serial* DWS theory. This approach keeps derivations of linguistic expressions that have the same main constituent structure much more uniform regardless of how many phrasal specifiers will end up being created.

There is another reason why this serial(ly bottom-up) approach is preferable. Again, let's assume that initiating DWSs increases complexity. If this is the case, the

system should be designed in such a way that DWSs be created only when needed to derive the correct constituent structure. Clearly, the parallel DWS theory violates this tenet, since nothing prevents a derivation like (11), for example (same LA as (10)), in which simple specifiers initiate their own DWSs, when one DWS alone would derive the correct constituent structure (recall (10)):

(11) LA = {they, believed, him, to, want, him, to, expect, her, to, be, mad, at, him}

DWS_1 = {they}, DWS_2 = {him}, DWS_3 = {him}, DWS_4 ={her}, DWS_5 ={him}

Interestingly, this could be taken to be an argument for the original formulation of Merge (that is, between two lexical items, between a lexical item and a phrase, or between two phrases) and against the idea that a lexical item alone can initiate a DWS as in Unary Merge or in my Merge(lexical item, Ø). This argument doesn't go through, however, in the light of examples like (12):

(12) LA = {$_D$the[N], $_N$men, $_T$will[V], $_V$arrive[D]}

DWS_1 = {$_D$the, $_N$men}, DWS_2 = {$_T$will, $_V$arrive[D]}

In (12), Merge($_D$the[N], $_N$men) and Merge($_T$will, $_V$arrive[D]) have applied, unnecessarily creating two DWSs. This derivation shows that preventing a single lexical item from initiating a DWS doesn't solve the problem that the parallel DWS approach has.

Whatever theory of Merge we choose, it seems that a system with parallel DWSs can always generate more DWSs than needed, as in (12).[15]

Let's assume then that all derivations start with a single DWS and the creation of additional DWSs is restricted to cases in which the lexical items remaining in the LAs wouldn't be able to be inserted otherwise (recall (2) above). Let's briefly illustrate the properties of this serial approach to DWSs with the following derivation:

(13) LA = {$_D$the[N], $_N$man, $_v$*v*[V][D], $_V$saw[D], $_D$her}

Target($_D$her) ⇒ DWS_1 = {$_D$her}

Target($_V$saw[D]) ⇒ DWS_1 = {$_V$saw, {$_D$her}}

Target($_v$*v*[V][D]) ⇒ DWS_1 = {$_v$*v***[D]**, {$_V$saw, {$_D$her}}}

Target($_v$*v*[D]) ⇒ (subroutine) Target($_N$man) ⇒ DWS_2 = {$_N$man}

Target($_D$the[N]) ⇒ DWS_2 = {$_D$the, {$_N$man}}

Target($_v$*v*[D]) ⇒ DWS_1 = {{$_D$the, {$_N$man}}, {$_v$*v*, {$_V$saw, {$_D$her}}}}

As can be seen in this derivation, an additional DWS is created only when the SelF of *v* needs to be satisfied. Before that, like when *saw* is inserted, for example, no additional DWS may be initiated, because no unsatisfied SelFs remain in the DWS. Consequently, only a new head that selects for the root category in the DWS can be inserted (and only in the same DWS). Notice that this approach observes Collins's (2002) Locus Principle, since the same head is being targeted in DWS_1 before and after DWS_2 is created (see the steps in bold in (13)). Actually, we could go even further and claim that the system

creates additional DWSs *in order to* obey the Locus Principle when phrasal specifiers need to be inserted.

Let's formalize this restriction regarding additional DWSs with the principle in (14):

(14) *Additional DWSs*

An additional DWS may be initiated only when Target α applies to a lexical item that is already inside a DWS.

This approach addresses the concern that under standard assumptions derivations involving complex specifiers proceed differently from derivations involving simple ones, since branching specifiers are built independently of the properties of the element whose SelFs they satisfy.[16] Under the approach I propose here, this is not the case; derivations that only differ with respect to whether their specifier is complex or simple are different only after the specifier's head is merged.

5.2. Inter-DWS Operations

We haven't formalized yet how Merge should proceed once an additional DWS is completed. In order to do so, let's first examine the relevant portion of the derivation in (13) above, repeated here as in (15) (notice that the LA has been reduced accordingly):

(15) LA = {$_{v}v$[V][D], $_{D}$the[D], $_{N}$man}

Target($_{v}v$[V][D]) ⇒ DWS$_1$ = {$_{v}v$[D], {$_{V}$saw, {$_{D}$her}}}

Target($_{v}v$[D]) ⇒ (subroutine) Target($_{N}$man) ⇒ DWS$_2$ = {$_{N}$man}

Target($_{D}$the[N]) ⇒ DWS$_2$ = {$_{D}$the, {$_{N}$man}}

Target($_{v}v$[D]) ⇒ DWS$_1$ = {{$_{D}$the, {$_{N}$man}}, {$_{v}v$, {$_{V}$saw, {$_{D}$her}}}}

As can be clearly seen in (15), an object created in DWS$_2$ satisfies the SelF that licensed this DWS$_2$ by merging with DWS$_1$. This means that SelFs are instrumental not only in allowing additional DWSs but also in determining the following Merge operation when they are completed. Let's call this process DWS reduction:

(16) *DWS Reduction*

After DWS$_n$ is completed, the resulting syntactic object is merged in DWS$_{n-1}$.

Again, we can see how this approach to additional DWSs can be couched nicely in terms of Collins's Locus Principle. When a derivation reaches a point at which a SelF inside a DWS needs to be satisfied, targeting the head containing the SelF allows for an additional DWS to be created. Likewise, when the additional DWS has been completed, the head containing the relevant SelF is targeted again for Merge. The end result is that additional DWSs are heavily constrained by the locus of Target α.

Notice, however, that (16) doesn't make much sense in the case of DWS$_1$. What actually happens when DWS$_1$ is completed is that the DWS is spelled out and linearized,

which I discuss in section 5.4. Before that, I provide an argument that Merge operations applying in different DWSs cannot always be parallel.

5.3. DWSs and Logical Order of Merge

An obvious implication of the strong serial approach that I'm following here is that there can be no simultaneous operations of Merge in a derivation. This contrasts with the parallel DWS approach in that nothing prevents simultaneous operations of Merge applying in different DWSs. Under this approach, each parallel DWS could in principle undergo structure-building operations independently of the other DWSs.

The derivation of expressions involving specifiers inside specifiers, however, shows that that cannot be the case, since they require a specific ordering of Merge operations *even between* DWSs. Consider, for example, the derivation for the DP *that guy's mom's book* at the point depicted in (17):

(17) LA (reduced to Ø from {$_{D}$that[N], $_{N}$guy, $_{D}$'s[N][D], $_{N}$mom, $_{D}$'s[N][D], $_{N}$book})

DWS_1 = {$_{D}$'s[D], {$_{N}$book}}, DWS_2 = {$_{D}$'s[D], {$_{N}$mom}}, DWS_3 = {$_{D}$that, {$_{N}$guy}}

Given what we have seen in the previous section, the final DP *that guy's mom's book* is derived by merging *that guy* as the specifier of *'s mom*, and then merging *that guy's mom* as the specifier of *'s book*. It is obvious from (17) then that if we have, say, three DWSs, the correct constituent structure is never going to obtain, *unless we order the sequence of*

Merge. Crucially, nothing is merging in DWS_1 = {$_D$'s[D], {$_N$book}} while Merge ({that, {guy}}, {'s[D], mother}) applies, which means that ordering of Merge, even outside a single DWS is a necessary property of the system. The existence of structures with specifiers inside specifiers then provides evidence that Merge operations can't always be simultaneous. A parallel system, then, must necessarily allow for both simultaneous and ordered Merge operations to take place.[17]

Moreover, (17) also shows that the parallel approach is potentially inefficient. Since different DWS are created independently of one another, nothing prevents DWS_1 and DWS_3 from merging together, excluding DWS_2. Crucially, if this happens, DWS_2 is left unmerged, meaning that the system created a DWS that can't become part of the structure under construction.

5.4. Spell-out: revising DWS Reduction

A final part of this theory that needs to be formalized is the relation between DWSs and spell-out. As we saw at the end of section 5, (16) can't be applied to all DWSs but only to additional DWSs. Let's then modify the principle of DWS Reduction in the following way:

(18) *DWS Reduction (revised version)*

After DWS*n* is completed:

a. If n≠1, the resulting syntactic object is merged in DWS_{n-1}.

b. If n=1, the resulting syntactic object is spelled out.

It should be noted that the theory that I'm proposing here is in principle compatible both with theories that assume phases and with theories that don't. As already mentioned in section 4 above, phase-based theories generally assume that LAs are composed of subarrays. If this is correct, we have to assume that DWS_1 is completed when no SelFs remain in the DWS and no lexical item remains in the relevant subarray. If we assume the architecture of early Minimalism (Chomsky 1995), we have to say instead that DWS_1 is completed when no SelFs remain in the DWS and no lexical item remains in the LA.

The case of purely derivational approaches is a bit more complex, since they assume that every operation is interpreted at the interfaces right after it applies (see e.g. Epstein et al. 1998, Epstein and Seely 2002b, 2006). In order to make my approach compatible with this, we could actually say that (16) (and not (18)) holds, since spell-out seems to be unrestricted and just understood as a consequence of Merge, rather than the result of reaching a certain point in a derivation. It could also be the case that, as Grohmann (2006) proposes, the interfaces interpret information provided by the structure-building operations at certain points in the derivation (what he calls Prolific Domains) and then spell-out applies at the end of the derivation.

Again, deciding between these theoretical options lies beyond the scope of this paper. The present proposal, however, seems to be compatible with all of them.

6. DWSs and Linearization

6.1. Implications for Linearization

In this paper I have been assuming, as is standard in Minimalism, that Merge creates sets of lexical items, without any representation of linear order. Let's explore now some important consequences of the approach that I have been arguing for here for the theory of linearization.

These consequences all stem from the fact that this serially bottom-up approach complements entails that complements are built before heads and heads are built before specifiers regardless of phrasal status. Moreover, elements inside specifiers are also inserted in the derivation after heads and complements.

It seems then that if the approach I'm proposing here is correct, the Kaynean Specifier-Head-Complement linear order can be derived by just reversing the order in which lexical items entered the derivation, as suggested by Fernández-Salgueiro (2002, 2007) and Zwart (2004, 2011). Linearization can be understood as a last-in, first-out procedure that has the expected computational properties of stacks. Following this idea, let's propose (19):

(19) *Stack-based Linearization*

Lexical items are linearized by reversing the order in which they were inserted in the derivation.

Below I explore in more detail this approach to linearization with respect to the

"symmetry at the right edge" problem, phrasal specifiers, linearization of chains, the Condition on Extraction Domains (CEDs), and cyclic linearization of syntactic structure.

6.2. Symmetry at the right edge

Under Kayne's (1994) Linear Correspondence Axiom (LCA) heads asymmetrically c-command (hence precede) the elements inside their complements. This is straightforward in X'-Theory given the existence of vacuous projections. Once bare phrase structure is adopted, however, a problem arises: the first operation of Merge, which concatenates two lexical items, always introduces a point of symmetry that can't in principle be linearized.

By assuming that DWSs can be initiated with just one lexical item, we can solve this problem (see also sections 3 and 4 above). As also pointed out by Zwart (2004, 2011) Kayne's idea that there's an inherent asymmetric relation between heads and complements can be easily captured under this type of approach, including Zwart's, Frampton and Gutmann's, and the present proposal.

One of the theoretical contributions of the approach I develop here then is that this asymmetric property of the system, which Chomsky's (1994) bare phrase structure made it hard to capture, is not incompatible with the notion that Merge is inherently binary. In this sense, the present study offers a synthesis of these two approaches.

6.3. Phrasal Specifiers: how many applications of Spell-out?

There is another important asymmetric property of syntactic structure that has been problematic for different approaches to linearization, which is the fact that phrasal

specifiers also precede heads and complements. For instance, Kayne himself was compelled to assume, without clear independent motivation, that specifiers are actually adjuncts, so that they could asymmetrically c-command the rest of the structure. When Chomsky (1995) adapted Kayne's theory to the bare phrase structure framework, he had to claim that intermediate projections do not c-command in order to avoid the same problem. This idea is problematic, since X' projections do count when c-command relations are computed, otherwise a complement would always c-command the specifier of the phrase it belongs to.

An added related problem is the fact that the lexical items inside the specifier precede the head and the complement when there's no syntactic relation at all between these elements. In order to deal with this issue, Kayne also proposes a recursive step of the LCA, which involves computing dominance relations as well. This is also assumed by Chomsky (1995).

Uriagereka (1999) rejects this kind of approach altogether and argues for a multiple spell-out model that derives Kayne's recursive step of the LCA. In his model, specifiers (and adjuncts) trigger a separate application of spell-out, which renders their internal structure inaccessible to syntactic operations related to elements outside their DWS (the basic idea behind Huang's (1982) CEDs). Once DWSs are assumed to lack phrasal specifiers, asymmetric c-command is enough to derive linear order for all terminals. This is a welcome result, especially given the fact that it removes the need for computing dominance relations in the PF component. This way computational complexity is reduced, not only in terms of the number of elements that has to be

computed, but also in terms of which kinds of elements, since only terminal elements are taken into account.

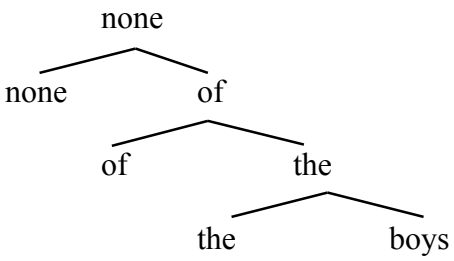
However, it is not clear what mechanism is responsible for deciding which DWS spell-out should apply to. Consider the situation in (20), for example:

(20)

If spell-out must apply to DWSs "prior to their merger" as Uriagereka (p.256) claims, what forces {the, boys} to undergo spell-out and prevents {v, {did, it}} from doing so? Uriagereka claims that only maximal projections undergo spell-out ({the, boys} in this case). The problem with this idea is that in Minimalism, projections are maximal until they project further (if they ever do). Therefore, under bare phrase structure, both DWSs are maximal projections at the point at which the system has to make a decision regarding spell-out.

Another, but related, potential problem is that at the point in the derivation illustrated in (20) there's no way to know whether more Merge operations will apply in DWS_1 or not. While it's certainly true that {the, boys} can't project any further, it could well be the input for further Merge operations (in the same DWS), as in (21):

(21)



A paradox arises here. If spell-out applies to a DWS when a maximal projection is reached, the system can't build a phrasal specifier with more than two lexical items. If spell-out doesn't apply to a DWS when a maximal projection is reached, this object could then be merged with the syntactic object in the other DWS, making the structure impossible to linearize (since dominance relations can't be computed under this approach). Notice that inspecting unsatisfied SelFs wouldn't help either, since as (21) shows lack of a SelF inside a DWS doesn't really preclude it from expanding. At the point of the derivation in (20), then, no decision regarding spell-out can be made by just examining the syntactic objects present in the derivation.

Notice, however, that the source of these problems is the parallel DWS theory that Uriagereka is assuming, precisely the kind of approach that I have been arguing against in this paper. In a way, these problems are actually the multiple spell-out version of the Kayne/Chomsky problems described above. Once the serial DWS theory is adopted, specifiers are always introduced in the derivation after heads and complements, so the need to spell out specifiers independently for linearization purposes disappears. Moreover, we can maintain the conceptual advantages of Uriagereka's approach: dominance relations don't need to be computed and there's no need to treat specifiers as adjuncts or

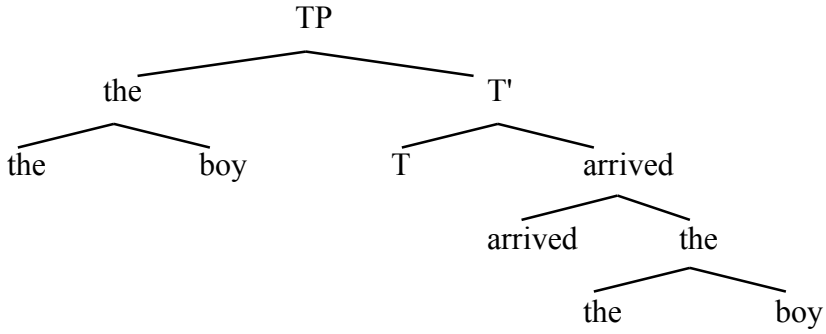
claim that intermediate projections don't c-command.

There is another interesting similarity between multiple spell-out and the approach that I am pursuing here. Uriagereka argues that economy considerations regulate the number of applications of spell-out. In his system, spell-out applies only if the structure at hand couldn't be linearized otherwise (due to the presence of complex specifiers). Likewise, I have argued that additional DWSs are also restricted by economy considerations, being created only when a phrasal specifier is to be introduced.

6.4. Linearization of chains

If we assume that linear order is the mirror image of order of Merge, questions arise regarding movement operations involving phrasal elements. Consider a simple unaccusative sentence like *the boy arrived*:

(22)



In this case, we would definitely predict that the specifier *the boy* precedes the rest of the sentence (since it was the last element to be merged), but how can we decide whether it's to be pronounced *the boy* or *boy the*? Since phrasal movement involves making a copy of

a complex element, this last Merge operation doesn't provide us with the necessary information.

A solution to this problem lies in the mechanism that is responsible for linearizing chains. Once trace theory is abandoned, linearization must be able to inspect movement chains in order to determine which copy (or copies) is phonetically realized. This is a necessary component of the linearization procedure, especially since it's been shown that the highest copy in a chain isn't always the element that is pronounced (see e.g. Nunes 1999, 2004 and Bošković 2002).

In the structure above, for instance, linearization first encounters the copy of the object {the, {boys}} and then inspects the chain's lower copy, which does have information regarding order of Merge. Since *the* was merged later than *boys* was, *the* is pronounced first.

A possible challenge for this approach comes from Epstein and Seely's (2006) claim that, even though chains are extensively referred to in our descriptions of syntactic phenomena, they can't be defined as syntactic objects. Moreover, the concept of chain is redundant with the concept of movement. Notice, however, that I'm claiming that chains are relevant in the PF component of the grammar, when linearization is to be computed. Thus, it could be the case that movement operations create chains that aren't visible to the narrow syntactic computation (if Epstein and Seely are right in that they aren't syntactic objects) but it doesn't preclude them from being visible in the PF component.

6.5. Accommodating CED effects

In the last eight years or so there has been a lot of discussion about the status of the CED. The traditional view is that subjects and adjuncts are islands for movement, which is also predicted under Uriagereka's multiple spell-out approach. However, it has been argued that sub-extraction from subjects is sometimes possible, though always more restricted than sub-extraction from objects. In English, for example, both pied-piping and preposition stranding are possible in the case of object sub-extraction, but only the former is possible in subject sub-extraction, as shown in (23):

(23) a. We talked about the car [of which]$_j$ [a picture e_j]$_i$ was taken e_i

b. *We talked about the car Op$_j$ [a picture of e_j]$_i$ was taken e_i

Examples like (23a) were used by Chomsky (2008) to argue that sub-extraction out of derived subject positions is possible, which he takes as evidence for phasal properties of derivations. Chomsky's arguments have been contested by Broekhuis (2006) and Boeckx (2008). Moreover, Gallego and Uriagereka (2006) have shown that availability of sub-extraction is related to the activity condition; it can only apply before case valuation.

Whatever the right analyses of these cases may be, the view that subjects are islands for extraction needs to be revised. The fact that sub-extraction from subjects is sometimes possible (even if more constrained than from objects) demands that the derivational structure-building procedure allow for it to ever take place, *contra* the traditional view. The architecture that I propose in this paper does in principle allow for

sub-extraction cases, since a phrasal specifier created in an additional DWS is merged in the DWS that licensed it (see (18a)) and thus aren't automatically turned into islands for movement as in Uriagereka (1999). For discussion of adjuncts, see section 7 below.

6.6. Cyclic linearization

Finally, I would like to point out that another advantage of my approach is that it is also compatible with Fox and Pesetsky's (2005) theory of cyclic linearization, under which the order of elements inside a phase is fixed after the phase is completed. If this approach turns out to be correct, it means that exhausting a subarray doesn't lead to spell-out, but to a point at which decisions regarding linear order must be made and kept constant throughout the rest of the derivation. Again, this can be easily translated to the present approach by modifying (18) slightly, as in (24) below:

(24) *DWS Reduction and cyclic linearization*

After DWS*n* is completed:

a) If n≠1, the resulting syntactic object is merged in DWS_{n-1}.

b) If n=1, linear order is computed (following (19)) and fixed.

Although in their paper Fox and Pesetsky argue that linear order is determined by the laws of precedence, they don't commit to any specific linearization algorithm, as the following quote illustrates:

> "We are agnostic as to the balance between language-specific and UG-determined aspects of these Laws" (Fox and Pesetsky 2005:40)

It seems then that the approach that I have developed here could be easily incorporated into Fox and Pesetsky's theory, once spell-out is redefined as "computing and fixing word order" (as in (24b)).

7. Some notes on Adjuncts

Notice that I haven't referred to adjuncts yet in this paper. The main reason for deferring the discussion of adjuncts is that their syntax is far from clear. The one thing that we seem to agree on, however, is that they don't get inserted in the derivation in the same way than arguments do.

When discussing the properties of Merge and LAs I capitalized on SelFs and argued that they were involved in triggering Merge operations and licensing additional DWSs. Adjuncts are definitely different in this respect, since they can become part of the structure of an expression without satisfying SelFs; adjuncts, by definition, are not required by other lexical items.

The fact that adjuncts are not required by SelFs correlates with the notion that adjuncts aren't inserted by means of Merge, but by an adjunction operation, which creates an ordered pair instead of a set (see e.g. Chomsky 1995). If it's true that there is a requirement on LAs based on SelFs, as I have argued here, it makes sense for adjunction to be treated differently. A reasonable conclusion then is that the lexical items that compose an adjunct form a separate LA and are merged in a separate DWS. Since there's

no SelF requiring Merge of the adjunct with the object in the other DWS, the only way to concatenate them is through an operation different from Merge.

By considering LAs on the one hand and DWSs on the other (two of the main notions I have discussed and formalized here) we can arrive at a three-way distinction between (phrasal) subjects, complements, and adjuncts, as shown in the following table:

(25)

	Separate LA?	Separate DWS?
Complements	NO	NO
Subjects	NO	YES
Adjuncts	YES	YES

According to (25), complements and subjects pattern together in that they satisfy SelFs (thus not triggering a separate LA).

Notice now that, as we saw in the case of subjects, being built in a separate DWS doesn't necessarily preclude sub-extraction, but it does mean that it's not as free. We are thus predicting that subjects and adjuncts pattern together with respect to sub-extraction, that is, generally disallowing it but permitting it under some extra conditions. In this respect, Truswell's (2007) has noted that sub-extraction is allowed from elements denoting a secondary predicate, as shown in (26) and (27) (contra Boeckx 2008 and Gallego and Uriagereka 2006, who claim that adjuncts always disallow sub-extraction):

(26) [What]$_i$ did John drive Mary crazy [trying to fix t_i]?

(27) [Which tune]$_i$ did John arrive [whistling t_i]?

We conclude then that subject sub-extraction is possible only if the subject's case features haven't been valued, while adjunct sub-extraction is possible only if the adjunct is somehow part of the event structure denoted by the main predicate. In both cases, then, we see that an extra condition tying subjects and adjuncts to the main spine is needed to permit sub-extraction; a syntactic one in the case of subjects and a semantic one in the case of adjuncts.

8. Conclusions

In this paper I have explored an approach to LAs, Merge, and DWSs under which SelFs play a major role. I have argued that both LAs and Merge are constrained by SelFs and that derivations start by targeting only one lexical item in a single DWS, making this approach computationally more efficient than the standard Minimalist approach. I have also explored several consequences for our understanding of the linearization procedure, and argued that linear order is derived from order of Merge. This approach provides novel solutions to old linearization-related problems, such as the symmetry at the right edge and the linearization of phrasal specifiers. Finally, I have shown that following this approach we can arrive at a correct characterization of the core properties of complements, specifiers, and adjuncts.

References

Boeckx, Cedric. 2008. *Bare syntax*. Oxford: Oxford University Press.

Bošković, Željko. 2002. On multiple wh-fronting. *Linguistic Inquiry* 33:351-383.

Broekhuis, Hans. 2006. Extraction from subjects: some remarks on Chomsky's 'On Phases', Ms., Universiteit Van Tilburg.

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

Chomsky, Noam. 2001. Derivation by Phase. In *Ken Hale: A Life in Language*, ed. by Michael Kenstowicz, 1-54. Cambridge, MA: MIT Press.

Chomsky, Noam. 2008. On Phases. In *Foundational Issues in Linguistic Theory. Essays in Honor of Jean-Roger Vergnaud*, ed. by Robert Freidin, Carlos Otero and Maria Luisa Zubizarreta, 133-166. Cambridge, MA: MIT Press.

Collins, Chris. 1997. *Local Economy*. Cambridge, MA: MIT Press.

Collins, Chris. 2002. Eliminating Labels. In Samuel Epstein and T. Daniel Seely, eds.

Epstein, Samuel, Erich Groat, Hisatsugu Kitahara, and Ruriko Kawashima. 1998. *A Derivational Approach to Syntactic Relations*. Oxford: Oxford University Press.

Epstein, Samuel and Norbert Hornstein, eds. 1999. *Working Minimalism.* Cambridge: Cambridge University Press.

Epstein, Samuel and T. Daniel Seely, eds. 2002a. Derivation and Explanation in the Minimalist Program. Oxford: Blackwell Publishers.

Epstein, Samuel and T. Daniel Seely. 2002b. Rule Applications as Cycles in a Level-Free Syntax. In Samuel Epstein and T. Daniel Seely, eds.

Epstein, Samuel and T. Daniel Seely. 2006. *Derivations in Minimalism*. Cambridge: Cambridge University Press.

Fernández-Salgueiro, Gerardo. 2002. Linearization and the Faculty of Language. MA Thesis, University of Vigo.

Fernández-Salgueiro, Gerardo. 2007. Reducing computation at the interface with the Sensory-Motor Systems: a derivational approach to (Chain) linearization. In *Proceedings of ConSOLE XIV*, ed. by Sylvia Blaho, Luis Vicente and Erik Schoorlemmer, 61-72.

Fortuny, Jordi. 2008. *The emergence of order in syntax*. Amsterdam: John Benjamins.

Fox, Danny and David Pesetsky. 2005. Cyclic linearization of syntactic structure. In *Theoretical Linguistics* 31:1-46.

Frampton, John and Sam Gutmann. 2002. Crash-Proof Syntax. In Samuel Epstein and T. Daniel Seely, eds.

Gallego, Ángel and Juan Uriagereka. 2006. Sub-extraction from Subjects. In *Romance Linguistics*, ed. by José Camacho, Nydia Flores-Ferrán, Liliana Sánchez, Viviane Déprez and María José Cabrera, 155-168. Amsterdam: John Benjamins.

Grohmann, Kleanthes. 2006. The Road to PF. In *Proceedings of the 17th International Symposium on Theoretical and Applied Linguistics*, ed. by Eleni Agathopoulou, Maria Dimitrikapoulkou and Despina Papadopoulou. Thessaloniki: Aristotle University of Thessaloniki.

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

Kitahara, Hisatsugu. 1997. *Elementary Operations and Optimal Derivations*. Cambridge, MA: MIT Press.

Nunes, Jairo. 1999. Linearization of chains and multiple realization of chain links. In Samuel Epstein and Norbert Hornstein, eds.

Nunes, Jairo. 2004. *Linearization of Chains and Sideward Movement*. Cambridge, MA: MIT Press.

Seely, T. Daniel. 2006. Merge, derivational c-command, and subcategorization in a label-free syntax. In *Minimalist Essays*, ed. by Cedric Boeckx, 182-218. Philadelphia: John Benjamins.

Truswell, Robert. 2007. Extraction from Adjuncts and the Structure of Events. *Lingua* 117:1355-1377.

Uriagereka, Juan. 1999. Multiple Spell-Out. In Samuel Epstein and Norbert Hornstein, eds.

Zwart, Jan-Wouter. 2004. Unary Merge. Paper presented at the Tilburg University Staff Seminar, October 28.

Zwart, Jan-Wouter. 2011. Structure and order: asymmetric merge. In *The Oxford Handbook of Linguistic Minimalism*, ed. by Cedric Boeckx, 96-118. Oxford: Oxford University Press.

[1] Throughout this paper, I will not be including the resulting label in the output of Merge. This is only for convenience, and I am actually abstracting away from whether the syntax needs labels or not. See Collins (2002) and Seely (2006) for discussion.

[2] Under the approach that Merge is not free but feature-driven, an additional problem arises, since this last Merge doesn't satisfy any SelF (see sections 3 and 4).

[3] Collins (1997) also claims that Merge isn't free, but applies to satisfy the principle of Integration.

[4] The actual term Unary Merge is from Zwart's (2011) and is defined in the following way: "Merge selects a single element from a resource and includes it in the object under construction." Frampton and Gutmann (2002) don't use the term Unary Merge but they do claim that a lexical item alone can initiate the derivation.

[5] Notice that this is just lexical information and doesn't refer to well-formedness conditions on Phrase Structure or the θ-Criterion.

[6] I'll be assuming that v is responsible for introducing an external argument.

[7] Chomsky (2001) suggests that Subarrays might be based on C and *v*.

[8] Not in the sense of Kitahara (1997) either.

[9] Taking seriously the idea that before any Merge operations the DWS is the empty set, it could be argued that only lexical items that don't display SelFs can initiate a derivation. Here I provide independent arguments to support this view.

[10] Again, I'm making the standard assumption that *v* is the element that selects for a DP as the external argument.

[11] This is actually another standard assumption in Minimalism. A head's SelFs must all be satisfied before new heads are introduced. Collins (2002) for example, dubs this the Locus Principle.

[12] Both Frampton and Gutmann's (2002) system and the one I proposed, however, are much more restricted since Merge is constrained by SelFs. Still, nothing prevents lexical items with no SelFs to initiate multiple parallel DWSs.

[13] Ignoring now functional categories, for simplicity.

[14] Interestingly, reducing computational complexity is taken to be the main purpose of subroutines in computer science.

[15] Notice that the derivation of the sentence *the men will arrive* only requires one DWS.

[16] Notice that this also means that the parallel approach treats DWSs in the same way regardless of whether specifiers are to be attached to a less embedded specifier or to the main spine of the tree.

[17] Actually, we could even take these structures as evidence suggesting that simultaneous Merge operations don't exist.