On the Definition of Merge

Erik Zyman · The University of Chicago · January 22, 2021

Abstract. Two of the fundamental tasks of syntactic inquiry are to identify the elementary operations that build syntactic structure and to determine precisely what properties they have and why. This paper aims to bring us closer to those goals by investigating Merge. Two recent formal definitions of Merge that are explicit about its relationship to selection are evaluated. It is argued that both have significant strengths but also some theoretical drawbacks—and, more broadly, that set-theoretic definitions of Merge in general run into conceptual problems, regardless of whether they incorporate projection or not. It is proposed that Merge is not set-theoretic but graph-theoretic in nature: the syntactic objects it operates on and creates are (Bare Phrase Structure–compliant) phrase structure trees. Two new formal definitions of Merge, understood as a graph-theoretic operation, are proposed, and their benefits and costs compared. One obeys the No-Tampering Condition, but at the cost of making it unclear why Merge(α,β) satisfies only one selectional feature of α, not all of them. The other makes that observation fall into place straightforwardly, but at the cost of narrowly violating the No-Tampering Condition. The larger picture that emerges is one in which Merge is a graph-theoretic rather than a set-theoretic operation.

Keywords. Merge; set theory; graph theory; formalization; No-Tampering Condition; theoretical syntax.

Many thanks, for valuable discussion, to Karlos Arregi, Dan Brodkin, Bob Freidin, Matt Hewett, Nick Kalivoda, Zach Lebowski, Jim McCloskey, Jason Merchant, Ross Rauber, Darragh Winkelman, and the participants in my seminars on "Elementary Syntactic Operations" (Autumn 2019) and "Merge" (Winter 2021). The usual disclaimers apply.

1. Introduction

Two of the fundamental tasks of syntactic inquiry are to identify the elementary operations that build syntactic structure and to determine precisely what properties they have and why. This paper aims to bring us closer to those goals by investigating in detail the structure-building operation Merge (Chomsky 1995a:chap. 4, 1995b, a.m.o.),[1] with a focus on the two related questions of what properties it has and precisely how it should be formalized.

The discussion will proceed as follows. Sections 2 and 3 examine two recent formal definitions of Merge that both have the significant advantage that they are explicit about the relationship between Merge and selection: Collins and Stabler's (2016) definition of Triggered Merge and Merchant's (2019) definition of Merge. Both of these definitions of Merge are presented and then shown in action by means of sample derivations. It is argued that, despite their significant strengths, both definitions also have some theoretical drawbacks. A broader conclusion that is reached is that set-theoretic definitions of Merge in general (not only the two just mentioned) run into conceptual problems—and that this is so whether they incorporate labeling/projection as part of Merge or not.

These arguments pave the way for section 4 to develop an alternative, graph-theoretic approach to Merge, on which the syntactic objects that Merge operates on and creates are (Bare Phrase Structure–compliant) phrase structure trees, not sets (see note 26 below for relevant references). Two graph-theoretic definitions of Merge are developed, and their benefits and costs are compared. One obeys the No-Tampering Condition, but at the cost of making it unclear why Merge(α,β) satisfies only one selectional feature of α, not all of them. The other makes that observation fall into place straightforwardly, but at the cost of a narrow violation of the No-Tampering Condition. Section 5 concludes the paper by noting that, whichever of these two definitions or whatever modification of one of them ultimately turns out to be correct, the larger picture that emerges from this investigation is one in which Merge is best understood as a graph-theoretic rather than as a set-theoretic operation. As should be clear, this paper is a work of theoretical syntax in the sense of Chametzky 1996 (sect. 0.2) (and Chametzky 1994:245, 1995:163, 2008:192, 2011:312, and

[1] For overviews of recent syntactic theorizing about Merge, see Fukui 2011; Epstein, Kitahara, and Seely 2015; and Collins 2017a.

Hornstein 2009:ix; see also Epstein 1999 and Chametzky 2000, a.m.o.).

2. Collins and Stabler's (2016) Definition of Triggered Merge

We will begin with the definition of Triggered Merge formulated by Collins and Stabler (2016) (henceforth C&S). In order to lay out and evaluate this definition, it will first be necessary to present some of the other formal definitions it depends on.

2.1 C&S's Formal Definitions: A Brief Recapitulation

Consider first C&S's definition of *lexical item token*:

(1) **C&S's Definition 5**

A *lexical item token* is a pair $\langle \text{LI}, k \rangle$ where LI is a lexical item[2] and k is an integer.

(Following C&S [p. 45], we will usually write the integer as a subscript for convenience—i.e., we will write "her$_4$" for "$\langle$her,4$\rangle$.") The definition of lexical item token in (1) makes it possible for C&S to define *syntactic object* as follows:

(2) **C&S's Definition 7**

X is a *syntactic object* iff

(i) X is a lexical item token, or

(ii) X is a set of syntactic objects.

With the definition of syntactic object in (2) in place, C&S initially define Merge as follows:

(3) **C&S's Definition 13**

Given any two distinct syntactic objects A, B, Merge(A,B) = $\{$A,B$\}$.

Merge as defined in (3) is a version of Simplest Merge, in the sense of Epstein, Kitahara, and Seely

[2]For the definition of *lexical item* assumed in (1), see C&S's Definition 2. We will reproduce from C&S only those definitions that are directly relevant to the present discussion.

2014, 2015, Collins 2017a, and references cited in those works. What is important here is that (3) makes no mention of selection—or, therefore, of the relationship between Merge and selection.

Later on, however, C&S propose a different version of Merge, which they call Triggered Merge. Triggered Merge is formalized in a way that makes the relationship between it and selection clear.

C&S's definition of Triggered Merge depends on the notion of *trigger features*, which they explain as follows:

(4) "We call the features involved in triggering Merge 'trigger features.' We assume that such features are to be identified with subcategorization features, EPP features, and OP features for movement to Spec,CP…" (C&S:62).

The notion of trigger features in (4) allows C&S to define *Triggers*, a class of syntactic functions. Triggers is defined as follows:

(5) **C&S's Definition 26**

Triggers is any function from each syntactic object A to a subset of the trigger features of A, meeting the following conditions:

(i) If A is a lexical item token with *n* trigger features, then Triggers(A) returns all of those *n* trigger features. (So when $n = 0$, Triggers(A) = $\{\}$.)

(ii) If A is a set, then A = $\{$B,C$\}$ where Triggers(B) is nonempty, and Triggers(C) = $\{\}$, and Triggers(A) = (Triggers(B)) – $\{$TF$\}$, for some trigger feature TF $\in$ Triggers(B).

(iii) Otherwise, Triggers(A) is undefined.

The definition of Triggers in (5) makes it possible for C&S to define Triggered Merge as follows:

(6) **C&S's Definition 27**

Given any syntactic objects A, B, where Triggers(A) $\neq \{\}$ and Triggers(B) = $\{\}$, Merge(A,B) = $\{$A,B$\}$.

Another component of C&S's system that will be important for our purposes is their labeling algorithm, which relies on a function they call Label. C&S define Label as follows:

(7) **C&S's Definition 28**

Label is a syntactic function from syntactic objects to lexical item tokens, defined in the following way:

(i) For all lexical item tokens LI, Label(LI) = LI.

(ii) Let W be a derivable workspace. If {A,B} is contained in W, and Triggers(A) is nonempty, then Label({A,B}) = Label(A).

By this point, we have reviewed almost all the definitions from C&S that will be important for the discussion to follow. All that remains is for us to discuss their notion of *derivation* and lay out a few of their definitions related to it. According to C&S (p. 46), "[a] derivation of a syntactic object is a series of stages that constructs a single syntactic object from some lexical item tokens." C&S define a *stage* in a derivation as follows:

(8) **C&S's Definition 10**

A *stage* is a pair S = ⟨LA,W⟩, where LA is a lexical array and W is a set of syntactic objects. We call W the *workspace* of S.

The definition of stage in (8) enables C&S to define the operation Select as follows:

(9) **C&S's Definition 12**

Let S be a stage in a derivation S = ⟨LA,W⟩.

If lexical token[3] A ∈ LA, then Select(A,S) = ⟨LA – {A},W ∪ {A}⟩.

This in turn makes it possible for C&S to define *derivation* as follows:

[3]C&S occasionally abbreviate "lexical item token" to "lexical token."

(10) **C&S's Definition 14**

A *derivation* from lexicon L is a finite sequence of stages $\langle S_1, \ldots, S_n \rangle$, for $n \geq 1$, where each $S_i = \langle LA_i, W_i \rangle$, such that

(i) For all LI and k such that $\langle LI, k \rangle \in LA_1$, $LI \in L$,

(ii) $W_1 = \{\}$ (the empty set),

(iii) for all i such that $1 \leq i < n$, either

(derive-by-Select) for some $A \in LA_i$, $\langle LA_{i+1}, W_{i+1} \rangle = \text{Select}(A, \langle LA_i, W_i \rangle)$, or

(derive-by-Merge) $LA_i = LA_{i+1}$ and the following conditions hold for some A,B:

(a) $A \in W_i$,

(b) either A contains B or W_i immediately contains B, and

(c) $W_{i+1} = (W_i - \{A,B\}) \cup \{\text{Merge}(A,B)\}$.

With these definitions in place, we can proceed to consider a concrete example of a C&S-style derivation.

2.2 A C&S-Style Derivation

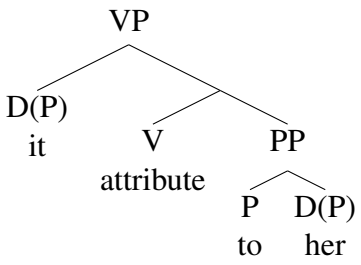
Consider the following sentence:

(11) I would attribute it to her.

On standard Larsonian assumptions (Larson 1988, 1990), the VP within (11) has the following structure (using tree notation for ease of exposition):[4]

[4]On the standard assumption that V undergoes head movement to a higher head *v* in English (Chomsky 1995a, a.m.o.; not shown in (12)), the correct constituent order will be derived (*I would* ***attribute it to her***, not **I would* ***it attribute to her***). The structure in (12) is probably too simple, for at least two reasons. First, McCloskey (2000:sect. 6.3) argues convincingly that, in sentences such as (11), the direct object DP actually originates lower than the PP, and comes to precede it by undergoing overt object shift (see Collins 2017b for a related investigation; see also Bošković 2002). Secondly, Harley (2008), Punske (2013), Koopman (2020), and Zyman (2020) argue on a number of grounds (see also Keyser & Roeper 1992) that Latinate prefixes such as *ad-*/*at-* in *attribute* are syntactically independent heads in the synchronic grammar of English, essentially identical to Germanic particles (compare ***I at-tribut****ed it to her diligence* with ***I chalk****ed it* ***up*** *to her diligence*). If all this is on the right track, then the structure of the "VP" (or $\sqrt{}$P) in (11) is closer to the following:

(12)



In C&S's system, the VP in (12) can be derived as follows. First, the lexical item tokens her_4 and to_3 are selected from the lexical array.[5] Once they are both in the workspace, they are eligible to be merged together:

(13) **Derivation of (12) in C&S's system (part 1)**

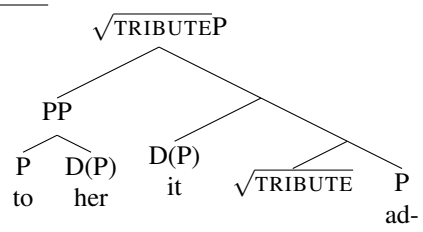
$$S_1 = \langle\{\text{attribute}_1, \text{it}_2, \text{to}_3, \text{her}_4\}, \{\}\rangle \rightarrow \text{Select her}_4$$

$$S_2 = \langle\{\text{attribute}_1, \text{it}_2, \text{to}_3\}, \{\text{her}_4\}\rangle \rightarrow \text{Select to}_3$$

$$S_3 = \langle\{\text{attribute}_1, \text{it}_2\}, \{\text{to}_3, \text{her}_4\}\rangle \rightarrow \text{Merge}(\text{to}_3, \text{her}_4)$$

Why are her_4 and to_3 eligible to undergo Triggered Merge? By hypothesis, her_4 bears no trigger features,[6] whereas to_3 bears one trigger feature: a structure-building feature [•D•], which requires that its bearer (or some projection thereof) merge with some syntactic object bearing the categorial feature [D].[7] More formally, Triggers(to_3) = {[•D•]}, and Triggers(her_4) = {}. Therefore, by the definition of Triggered Merge ((6)), Merge(to_3, her_4) is defined: it is $\{\text{to}_3, \text{her}_4\}$.[8]

(i)



The discussion in the text will keep to the simplified structure in (12) for ease of exposition.

[5]The integers are chosen arbitrarily for the purposes of the sample derivation and could be different without affecting anything.

[6]In the context of C&S's system, it seems reasonable to assume that, if her_4 bears a Case feature, this feature does not count as a trigger feature (see (4)).

[7]The bullet notation for structure-building features is from Heck and Müller 2007. The use of structure-building features in the text amounts to a slight elaboration of C&S's system, since C&S do not focus on the theory of features (p. 44). Important questions arise here about what the correct theory of features is and how it is to be formalized, but the sketch of an approach to structure-building features given in the text will suffice for the purposes of the derivation currently being laid out.

[8]According to the definition of Triggered Merge in (6), Merge(to_3, her_4) is defined simply because to_3 bears one

Since to_3 bears one trigger feature, and—by clause (ii) of the definition of Triggers ((5)), the set $\{to_3, her_4\}$ of which to_3 is a member bears one fewer trigger feature than to_3 itself does—$\{to_3, her_4\}$ bears no trigger features. That is, Triggers($\{to_3, her_4\}$) = $\{\}$.

We have just determined the value of Triggers($\{to_3, her_4\}$), but what is the value of Label($\{to_3, her_4\}$)? By clause (ii) of the definition of Label ((7)), Label($\{to_3, her_4\}$) = Label(to_3) (because to_3 is the member of $\{to_3, her_4\}$ that bears one or more trigger features—i.e., "what selects projects"). By clause (i) of the definition of Label, Label(to_3) = to_3. Therefore, Label($\{to_3, her_4\}$) = to_3.

As we have seen, then, to_3 and her_4 are indeed eligible to undergo Triggered Merge. After this operation applies, $attribute_1$ is selected and merged with $\{to_3, her_4\}$:

(14) **Derivation of (12) in C&S's system (part 2)**

$$S_4 = \langle\{attribute_1, it_2\}, \{\{to_3, her_4\}\}\rangle \rightarrow \text{Select } attribute_1$$

$$S_5 = \langle\{it_2\}, \{attribute_1, \{to_3, her_4\}\}\rangle \rightarrow \text{Merge}(attribute_1, \{to_3, her_4\})$$

Triggers($attribute_1$) = {[•to•],[9] [•D•]}. Triggers($\{to_3, her_4\}$) = $\{\}$, as determined above. Therefore, Merge($attribute_1$, $\{to_3, her_4\}$) is defined: it is $\{attribute_1, \{to_3, her_4\}\}$.

By the definition of Triggers ((5)), $\{attribute_1, \{to_3, her_4\}\}$ bears one fewer trigger feature than $attribute_1$ itself does (i.e., than the member of that set that bears one or more trigger features). The definition of Triggers itself does not tell us which of the two trigger features of $attribute_1$ is *not* inherited by the syntactic object that immediately contains it. But we know what the answer should be: [•to•] should not be inherited, since, intuitively, this trigger feature of $attribute_1$ has already triggered the merger of $attribute_1$ with a *to*-PP. Thus, although the current version of C&S's system does not establish unambiguously whether Triggers($\{attribute_1, \{to_3, her_4\}\}$) = {[•to•]} or {[•D•]}, we will take it that it is the latter (clearly the desired result[10]) and continue the derivation

or more trigger features, whereas her_4 does not. This definition, then, does not yet capture our theoretical intuition that Merge(to_3, her_4) is defined because the trigger feature of to_3 (its structure-building feature [•D•]) is *matched* by a feature of her_4 (namely, its categorial feature). This issue, though important, will be set aside here, since it is not directly relevant to the discussion to follow.

[9]This trigger feature is [•to•] rather than, e.g., [•P•] because the V *attribute* lexically selects (L-selects) the P *to* specifically (see Pesetsky 1991:chap. 1, Merchant 2019): compare *I attributed it* {***to***/****on***} *her* with *I blamed it* {***on***/****to***} *her*.

[10]Slightly elaborating C&S's formalization to actually guarantee this result should be straightforward, but that task

with that assumption in place.

Triggers({attribute$_1$, {to$_3$, her$_4$}}), then, = {[•D•]}. And, by the definition of Label ((7)), Label({attribute$_1$, {to$_3$, her$_4$}}) = Label(attribute$_1$) = *attribute*$_1$.

Next, *it*$_2$ is selected from the lexical array and merged with {attribute$_1$, {to$_3$, her$_4$}}:

(15) **Derivation of (12) in C&S's system (part 3)**

S_6 = ⟨{it$_2$}, {{attribute$_1$, {to$_3$, her$_4$}}}⟩ → Select it$_2$

S_7 = ⟨{}, {it$_2$, {attribute$_1$, {to$_3$, her$_4$}}}⟩ → Merge(it$_2$, {attribute$_1$, {to$_3$, her$_4$}})

Since Triggers({attribute$_1$, {to$_3$, her$_4$}}) = {[•D•]}, and Triggers(it$_2$) = {} (by hypothesis), these two syntactic objects are indeed eligible to undergo Triggered Merge, yielding {it$_2$, {attribute$_1$, {to$_3$, her$_4$}}}:

(16) **Derivation of (12) in C&S's system (part 4)**

S_8 = ⟨{}, {{it$_2$, {attribute$_1$, {to$_3$, her$_4$}}}}⟩

Since Triggers({attribute$_1$, {to$_3$, her$_4$}}) = {[•D•]}, and the newly formed syntactic object that immediately contains it bears one fewer trigger feature (by the definition of Triggers in (5)), Triggers({it$_2$, {attribute$_1$, {to$_3$, her$_4$}}}) = {}.

And by the definition of Label ((7)), the label of this newly formed syntactic object is identical to the label of the syntactic object it immediately contains that bears one or more trigger features. That is, Label({it$_2$, {attribute$_1$, {to$_3$, her$_4$}}}) = Label({attribute$_1$, {to$_3$, her$_4$}}) = *attribute*$_1$.

This concludes our C&S-style derivation. Now that the workings of C&S's system have been shown in action, we can proceed to evaluate their definition of Triggered Merge—and related components of their system, such as Label—on theoretical grounds.

will not be undertaken here, since it will not be directly relevant to the discussion to follow.

2.3 Evaluating C&S's Definition of Triggered Merge and Their Functional Approach to Trigger Features and Labeling/Projection

This section will discuss some issues that arise in connection with C&S's definition of Triggered Merge. The discussion will focus on C&S's functional approach to trigger features and labeling/projection (which relies on the functions Triggers and Label). It will be argued that, despite its theoretical virtues, the functional approach runs into several conceptual problems, which are briefly summarized here to foreshadow the discussion to follow. First, there is no theory of what is a possible syntactic function, so it is very difficult for the functional approach to account for the nonexistence of pathological syntactic functions. Secondly, the admission of syntactic functions into the system effectively bloats it from being a mixed derivational/representational system (which is arguably unproblematic in itself) to being a mixed derivational/representational/*functional* system (i.e., it bloats the syntactic ontology that the system relies on). Third, there is a worrying redundancy between the functions and the representations. It will also be argued that having every application of Merge(A,B) = {A,B} immediately "potentiate" (i.e., cause to be defined) Triggers({A,B}) and Label({A,B}) has the consequence that C&S's syntax is essentially three syntaxes running in parallel. Let us take up each of these points in turn.

One important property of Triggered Merge, as formulated by C&S, is that it is a pure structure-building operation: it assembles larger syntactic objects out of smaller syntactic objects, but does not incorporate a suboperation that satisfies features (e.g., trigger features), or a suboperation of labeling/projection. In fact, C&S's system does not incorporate feature satisfaction (or feature checking, or feature discharge) as such at all (p. 64), though this operation finds an approximate equivalent in C&S's system in the Triggers function that they posit. As for labeling, this is carried out in C&S's system not by Triggered Merge but by the Label function (in this, they follow Collins 1997:64). In other words, although the negotiation of trigger features and the work of labeling could in principle be assigned to Merge (as is the case on some other approaches to Merge: see section 3 below), this is not the case in C&S's system: instead, this work is "outsourced" to the syntactic functions Triggers and Label.

C&S's system, then, "unbundles" Merge (Collins 2017a): it dissociates a number of syntactic processes that could in principle all be suboperations of Merge—namely, structure-building, the negotiation of trigger features, and labeling—and assigns the latter two to separate syntactic functions (Triggers and Label respectively). All else being equal, this type of unbundling of syntactic operations—what Freidin (2016) refers to as avoiding the compounding of elementary operations—is theoretically highly desirable from a minimalist perspective, since it seems to make it possible to dispense with components of the theory that are suspiciously complex (e.g., an unwieldy, multipartite definition of Merge), replacing them with more elementary, more minimal operations (and/or functions).[11] In this case, however, all else is not equal: the introduction of the functions Triggers and Label into the theory (on which C&S's unbundling of Merge relies) gives rise to several conceptual problems.

First, there is no theory of what is a possible syntactic function. C&S's system is one in which the theory of syntax makes available not only derivational operations (Triggered Merge, Select, and Transfer)—which are functions involved in mapping one derivational stage to the next—but also syntactic functions that are *not* directly involved in mapping one derivational stage to the next, namely Triggers and Label. (We will continue to refer to the former as *operations* and to the latter as *functions*, following C&S.) Given that C&S's system does not impose any constraints on possible functions, it would, in principle, be perfectly possible in their system to posit the following function (which is closely modeled after Label, as formalized in (7)):

[11] See Hornstein and Nunes 2008, Hornstein 2009, and Larson 2014 for a different approach to unbundling Merge—one on which familiar cases of "Merge" actually involve the application of two separate, more elementary operations Concatenate and Label.

(17) **Nonhead (hypothetical function)**

Nonhead is a syntactic function from syntactic objects to lexical item tokens, defined in the following way:

(i) Let W be a derivable workspace. If {A,B} is contained in W, and Triggers(A) is nonempty, then Nonhead({A,B}) = Label(B).

(ii) Otherwise, Nonhead({A,B}) is undefined.

Clause (i) of the definition of Nonhead ((17)) is identical to clause (ii) of the definition of Label ((7)) except that "Label(A)" has been replaced with "Label(B)." That being so, for a syntactic object {A,B} such that Triggers(A) is nonempty (informally: such that A is a selector) and, therefore, Triggers(B) = {}, Nonhead({A,B}) returns the label of B (the nonhead), not the label of A (the "head" or selector). If the hypothetical function Nonhead were indeed made available by the theory of syntax, then, just as K = {it_2, {$attribute_1$, {to_3, her_4}}} has certain properties in virtue of the fact that Label(K) = $attribute_1$ (in C&S's terms)—namely, that it is a verbal rather than a nominal constituent (a "VP"[12]), with all the distributional and selectional consequences that that has—so too would we expect K to have syntactically relevant properties in virtue of Nonhead(K) being equal to it_2. But this is a pathological prediction: a VP such as {it_2, {$attribute_1$, {to_3, her_4}}} that contains (crucially, not as its head) a nominal object of V such as it_2 does not behave differently, in syntactically significant ways, than does a VP containing no nominal objects of V, such as {$sing_5$, {to_6, her_7}} (as in *She likes to sing to her*). Thus, the two types of VPs occur in the same region of clause structure and as complements to the same type of lexical item (v). Indeed, these similarities are why {it_2, {$attribute_1$, {to_3, her_4}}} and {$sing_5$, {to_6, her_7}} are both taken to be VPs in the first place—or, differently put, why syntactic objects are taken by many to be endocentric.

Examples of hypothetical syntactic functions such as Nonhead that are logically possible but empirically pathological (in view of the actual properties of syntax) could easily be multiplied. But

[12] In what follows, the scare quotes around *VP* will generally be dropped, even though the notion of VP is not an official part either of C&S's system or of other approaches to syntax that incorporate the main insights of Bare Phrase Structure (Chomsky 1995b).

suffice it to say that, once syntactic functions (as distinguished from derivational operations in the sense mentioned above) are admitted into the theory, there appears to be no deep and principled way to account for the fact that pathological syntactic functions do not exist—either instead of or in addition to nonpathological functions such as the putative functions Triggers and Label. By contrast, if syntax simply does not make use of syntactic functions (as distinguished from derivational operations) in the first place, then the nonexistence of pathological functions follows trivially.

It might be objected that, if we respond to the pathological-functions problem by eliminating freestanding syntactic functions from the theory and restricting ourselves to derivational operations, this will probably require some "un-unbundling" or "rebundling"—i.e., complicating the definitions of operations such as Merge to take over the work of the eliminated syntactic functions—and then the pathological-functions problem will reappear in a different guise. In particular, we will still be faced with the question of why the rebundled definition of Merge incorporates a substep or suboperation whose effects are similar to those of the eliminated function Label, but not a suboperation whose effects are similar to those of Nonhead. In other words, we will still be faced with the deep question of why syntactic structures are endocentric—or, differently put, why, for a syntactic object M = {A,B} with head/selector A and nonhead/selectee B, only A "counts," and B does not "count" at all, for purposes of determining the categorial and distributional properties of M. The point is well taken. Even so, though, a function-free approach to syntax seems to be on at least slightly better footing than an approach to syntax that incorporates functions, all else being equal. Both C&S's system, which incorporates functions, and function-free systems of the sort being defended here (of which two in particular will be developed in detail in section 4) incorporate derivational operations—i.e., the approaches to syntax under consideration here are all derivational. For every putative derivational operation, the Minimalist Program urges us to ask why it exists and why it has the properties it does. So it is true that, if a rebundled definition of Merge ends up incorporating a suboperation with effects similar to those of Label but not one with

effects similar to those of Nonhead, we will still have to ask why that is.[13] Nonetheless, whereas a function-free (but still derivational) approach to syntax obligates us to ask those difficult why-questions about every derivational operation, an approach to syntax that incorporates functions obligates us to ask those why-questions about every derivational operation *and* every syntactic function, which plausibly complicates the explanatory task.

These considerations bring us directly to the second conceptual problem we face if we introduce functions into the theory of syntax: doing so complicates the syntactic ontology that the theory is based on. If C&S's system did not incorporate functions, it would be a mixed derivational/representational system: it is partially derivational because it relies crucially on derivations for structure-building (see C&S's formal definition of derivation in (10)), and it is partially representational because the derivations give rise to representations (e.g., $\{it_2, \{attribute_1, \{to_3, her_4\}\}\}$) that undergo further syntactic operations, such as Triggered Merge and Transfer. To the mixed derivational/representational nature of C&S's system, no objections will be raised here. What is important for present purposes is that the introduction of functions into the system bloats it from being a mixed derivational/representational system to being a mixed derivational/representational/*functional* system—i.e., it bloats the syntactic ontology undergirding the approach. This is so because C&S's syntactic functions (Triggers and Label) are neither derivational operations (i.e., operations directly involved in mapping one derivational stage to the next) nor representations. Rather, they are a third kind of thing: functions that are *not* directly involved in mapping one derivational stage to the next.

That introducing functions into the system bloats the latter from being a mixed derivational/representational system to being a mixed derivational/representational/*functional* system, complicating the syntactic ontology in the manner just described, in itself motivates a search for alternative approaches to syntax that do not incorporate functions. In fact, though, the introduction of func-

[13] Perhaps there is pressure against derivational operations being too complex, and, as a result, Merge can incorporate an approximate equivalent of Label or of Nonhead, but not approximate equivalents of both—and the former is favored (i.e., "what selects projects" wins out over "what doesn't select projects") for reasons that remain to be elucidated. By contrast, if syntactic functions exist, it is less clear that the inventory thereof is under such intense pressure to be as small as possible (plausible though it is to suppose that there is some such pressure) that the existence of the function Label, which by hypothesis is preferred, effectively precludes the existence of the hypothetical function Nonhead.

tions into the theory is conceptually problematic for another reason, one that goes beyond the observation that it causes the ontological bloat just mentioned. There is an odd and theoretically worrying redundancy between the functions and the representations. Holding constant C&S's assumption that syntax is at least partially derivational, and also their assumption that syntax is nonetheless also partially representational, we naturally expect the linguistically significant properties of the syntactic objects derived to be recorded in the representations.[14] Unfortunately, on C&S's approach, two syntactically very significant properties of a syntactic object SO are not recorded in the representation of SO: its label and the specification of its trigger features (see Lasnik & Stone 2020:204 for a similar point). Instead, the representation of SO is the result of pure structure-building (recursive set formation). Although representations of this sort might initially appear attractively simple, and therefore appealing from a minimalist perspective, their simplicity is something of an illusion—one made possible by "hiding" the trigger features and the labels of the syntactic objects in what are essentially other dimensions (by recording them in the outputs of the functions Triggers and Label rather than in the representations themselves, where one would expect to find such syntactically significant properties of syntactic objects). In C&S's system, then, it is not the case that a *single* structure-building operation Merge[15] (along with Select) applies recursively to create larger and larger syntactic objects whose representations record all their linguistically significant properties. Rather, some of these properties—the syntactic objects' labels and trigger features—are computed (by the functions Label and Triggers, respectively) in parallel with the applications of Merge that build the syntactic objects up. Since every application of Merge(A,B) (yielding {A,B}) has as an immediate consequence that Triggers({A,B}) and Label({A,B}) come to be defined (i.e., to put it impressionistically, Merge(A,B) = {A,B} "potentiates" Triggers({A,B}) and Label({A,B})), it is not much of an exaggeration to say that C&S's syntax is really three syntaxes running in parallel.

[14] If this expectation were somehow shown empirically not to be met, then it would be quite unclear why representations even existed.

[15] We will follow C&S (p. 43) in setting aside Adjoin/adjunction (which may be an operation distinct from Merge: Chametzky 2000:142; 2003:sect. 5.3.2.3; Merchant 2014:3; Rauber 2019; for a different approach, see Hornstein & Nunes 2008, Hornstein 2009:chap. 4, and Larson 2014).

Summarizing, then, C&S define Triggered Merge as a pure structure-building operation, and outsource labeling and the negotiation of trigger features to two syntactic functions distinct from the operation Triggered Merge—namely, Label and Triggers respectively. Although this approach initially seems quite attractive from a minimalist perspective owing to its faithful adherence to the logic of unbundling, its reliance on a functional approach to trigger features and labeling runs into a number of conceptual problems:

(18) a. There is no theory of what is a possible syntactic function, and hence no principled account of the nonexistence of pathological functions such as Nonhead ((17)).

b. The functional approach bloats the system from being a mixed derivational/representational system to being a mixed derivational/representational/*functional* system (i.e., it bloats the syntactic ontology that the system relies on).

c. There is a worrying redundancy between the functions and the representations: some syntactically significant properties of syntactic objects are, contrary to expectation, not recorded in their representations but in the outputs of the functions (i.e., "hidden" in other dimensions), robbing the representations of much of their raison d'être.

d. Every application of Merge(A,B) = {A,B} immediately "potentiates" (i.e., causes to be defined) Triggers({A,B}) and Label({A,B}), with the consequence that C&S's syntax is essentially three syntaxes running in parallel.

In the following section, we will turn our attention to a different definition of Merge that avoids some of these problems but, as we will see, has certain theoretical drawbacks of its own.

3. Merchant's (2019) Definition of Merge

Merchant (2019:326, (2)) proposes a formal definition of Merge that is similar to C&S's in that it is explicit about the relationship between Merge and selection, but differs from C&S's in various other respects:

(19) **Merge(α, β)**

For any syntactic objects α, β, where α bears a nonempty selectional list $\ell = \langle \bullet F_1, \ldots, \bullet F_n \rangle$ of selectional features, and β bears a categorial feature F' that matches $\bullet F_1$,

call α the head and

a. let $\alpha = \{\gamma, \{\alpha - \ell, \beta\}\}$ call γ the *projection* of α, and

b. if $n > 1$, let $\ell = \langle \bullet F_2, \ldots, \bullet F_n \rangle$, else let $\ell = \varnothing$, and

c. let $\gamma = [\ \text{CAT}\ [cat(\alpha)]$
$\text{SEL}\ \ [\ell]\ \ ]$

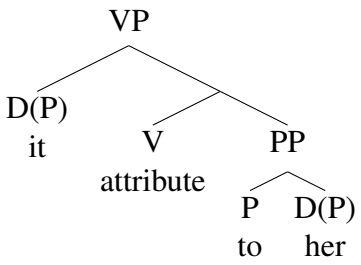
This version of Merge is shown in action in the following subsection.

3.1 A Merchant 2019–Style Derivation

Consider once more the sentence in (20), whose VP has the structure in (21) (= (12)), again using tree notation for ease of exposition:

(20) I would attribute it to her. (= (11))

(21)



In Merchant's (2019) system, the VP in (21) is derived as follows.[16] We take the lexical items *to* and *her* to have the following (partial) lexical entries:

[16]Merchant's (2019) definition of Merge ((19)) is embedded in a more extensive formalization of some of the core concepts of Minimalist syntax (e.g., the concept of derivation, among many others), which is laid out in Merchant 2014. The text will not reproduce any other formal definitions from Merchant's system (for which the interested reader is referred to Merchant 2014), partially because the system developed somewhat in the transition from Merchant 2014 to Merchant 2019 (e.g., the definition of Merge was revised somewhat), and partially because reproducing other formal definitions from Merchant's system will not be crucial to showing exactly how the definition of Merge in (19) works.

(22) a. $\text{to}_{[\text{CAT to, SEL } \langle \bullet \text{D} \rangle]}$

b. $\text{her}_{[\text{CAT D, SEL } \varnothing]}$

That is, $\textit{her}_{[\text{CAT D, SEL } \varnothing]}$ is an element of category D with an empty selectional list, and $\textit{to}_{[\text{CAT to, SEL } \langle \bullet \text{D} \rangle]}$ is an element of category *to*[17] whose selectional list bears one selectional feature: •D, which requires that its bearer merge with an element bearing the categorial feature [CAT D]. Because $\textit{her}_{[\text{CAT D, SEL } \varnothing]}$ bears a categorial feature that matches the first selectional feature on *to*'s selectional list, Merchant's definition of Merge ((19)) has the consequence that Merge($\text{to}_{[\text{CAT to, SEL } \langle \bullet \text{D} \rangle]}$, $\text{her}_{[\text{CAT D, SEL } \varnothing]}$) is defined (with $\text{to}_{[\text{CAT to, SEL } \langle \bullet \text{D} \rangle]} = \alpha$ in (19) and $\text{her}_{[\text{CAT D, SEL } \varnothing]} = \beta$).

By the definition of Merge in (19), the output of that Merge operation is computed as follows. By clause (a), the selector $\alpha = \textit{to}_{[\text{CAT to, SEL } \langle \bullet \text{D} \rangle]}$ is replaced by a more complex syntactic object $\{\gamma, \{\alpha - \ell, \beta\}\}$. $\alpha - \ell$ = the original selector α stripped of its selectional list ℓ—i.e., $\textit{to}_{[\text{CAT to, SEL } \varnothing]}$, which, for convenience, we can abbreviate $\textit{to}_{[\text{CAT to}]}$. (Here and below, we will occasionally suppress the selectional [= SEL] specifications of any syntactic object whose selectional list is the empty list [or 0-tuple] $\varnothing$.) Since ℓ at this point bears only a single selectional feature, clause (b) of (19)—in particular, the "else"-subclause—has the consequence that ℓ itself is replaced by the empty list $\varnothing$. By clause (c) of (19), γ is a feature structure consisting of an array (cf. Merchant 2014:sect. 1) of categorial features and an array of selectional features. The categorial array contains the categorial features of $\alpha = \textit{to}$ (i.e., the single categorial feature [CAT to]), and the selectional array contains ℓ, which is now equal to $\varnothing$. The result is that $\{\gamma, \{\alpha - \ell, \beta\}\}$—the complex syntactic object that replaced the original α—is the following (call it X for convenience):

(23) $\{[\text{CAT to, SEL } \varnothing], \{\text{to}_{[\text{CAT to}]}, \text{her}_{[\text{CAT D}]}\}\}$

[17] That is, the categorial feature of *to* is in fact not [CAT P] but rather simply [CAT to], so selection for *to* (an instance of L-selection) is formally completely parallel to selection for an element bearing, e.g., [CAT D] (an instance of c-selection, if that exists). If indeed *to* and other adpositions do not bear a categorial feature [CAT P], then we can account for Donca Steriade's observation (reported in Pesetsky 1991:9–10) that, although some verbs L-select particular adpositions, and others s-select locative or directional expressions, no verbs simply select a PP but are indifferent to the identity of its head. On Merchant's approach, which is also adopted by Hewett (2020:n. 3), c-selection for PP is impossible, because the categorial feature [CAT P] does not exist. Alternative approaches to accounting for Steriade's observation are certainly conceivable, but that is not directly relevant here.

We take the V *attribute* to have the following partial lexical entry:

(24) $\text{attribute}_{[\text{CAT V, SEL } \langle \bullet\text{to, } \bullet\text{D} \rangle]}$

That is, *attribute* bears the categorial feature [CAT V] and the ordered selectional list $\langle \bullet\text{to, } \bullet\text{D} \rangle$. At this point, we want to derive the result that $\textit{attribute}_{[\text{CAT V, SEL } \langle \bullet\text{to, } \bullet\text{D} \rangle]}$ can merge with X (= (23)). This will be possible only if X bears a categorial feature matching the first selectional feature on the selectional list of $\textit{attribute}_{[\text{CAT V, SEL } \langle \bullet\text{to, } \bullet\text{D} \rangle]}$, namely •to. X immediately contains a feature bundle (referred to as γ above) within which there is a categorial feature [CAT to], which does indeed match •to on $\textit{attribute}_{[\text{CAT V, SEL } \langle \bullet\text{to, } \bullet\text{D} \rangle]}$. That is, [CAT to] is present within the feature bundle γ, and $\gamma \in$ X. If, in this situation, X can be said to *bear* [CAT to], then $\textit{attribute}_{[\text{CAT V, SEL } \langle \bullet\text{to, } \bullet\text{D} \rangle]}$ will indeed be able to merge with X. We will assume here that this is indeed the case.[18]

Continuing with the derivation, then, $\textit{attribute}_{[\text{CAT V, SEL } \langle \bullet\text{to, } \bullet\text{D} \rangle]}$ merges with X (= (23)). By the definition of Merge in (19), this Merge operation yields the following (call it Y for convenience):

(25) $\{[\text{CAT V, SEL } \langle \bullet\text{D} \rangle], \{\text{attribute}_{[\text{CAT V}]}, \{[\text{CAT to, SEL } \varnothing], \{\text{to}_{[\text{CAT to}]}, \text{her}_{[\text{CAT D}]}\}\}\}\}$

We take the lexical item *it* to have the following partial lexical entry:

(26) $\text{it}_{[\text{CAT D, SEL } \varnothing]}$ (abbreviated as follows: $\text{it}_{[\text{CAT D}]}$)

In (25), the selectional list $\langle \bullet\text{D} \rangle$ occurs within a feature bundle that is immediately contained in Y. Assuming (in line with the discussion above) that, in this circumstance, Y can be said to *bear* the selectional list $\langle \bullet\text{D} \rangle$, it will be possible to merge Y with $\textit{it}_{[\text{CAT D}]}$, because the latter bears the categorial feature [CAT D], which matches the first (and only) selectional feature on the selectional

[18] Thanks to Karlos Arregi for helpful discussion of this issue. We will not develop here the formal definitions that would be needed to guarantee this result in Merchant's (2014, 2019) system (i.e., definitions of *bear* and related concepts), because doing so will not be strictly necessary for the discussion to follow.

Arregi also points out that, although the definition of Merge(α,β) in (19) is explicit about what happens to α (it is replaced by the more complex object $\{\gamma, \{\alpha - \ell, \beta\}\}$), it is not explicit about what happens to the original operand β. This issue should be resolvable by embedding (19) in a network of formal definitions of *workspace* and related notions, but that task too is set aside here as not strictly necessary for present purposes.

list borne by Y, namely •D. By the definition of Merge in (19), Merge(Y, $\text{it}_{[\text{CAT D}]}$) yields the following syntactic object:[19]

(27) {[CAT V, SEL ∅], {$\text{it}_{[\text{CAT D}]}$, {[CAT V], {$\text{attribute}_{[\text{CAT V}]}$, {[CAT to, SEL ∅], {$\text{to}_{[\text{CAT to}]}$, $\text{her}_{[\text{CAT D}]}$}}}}}}

This concludes the derivation of the VP in (21) in Merchant's (2019) system.

3.2 Evaluating Merchant's (2019) Definition of Merge

Now that we have seen in detail how Merchant's (2019) definition of Merge works, we can proceed to evaluate it from a theoretical standpoint. Two (related) advantages that this definition of Merge has over C&S's definition are that it does not rely on a syntactic function Triggers and that it does not require the postulation of a syntactic function Label.[20] We will return to the advantages (and drawbacks) of Merchant's approach to projection below.

This section is organized as follows. First, it will be shown that Merchant's definition of Merge violates the No-Tampering Condition (Chomsky 2005:11, 2007:8), and it will be argued on conceptual grounds that this should be avoided if possible. Secondly, it will be argued that Merchant's set-based approach to projection (a modification of the approach developed in Chomsky 1995b) has theoretical drawbacks that suggest that we should instead pursue a different approach to projection—and, by extension, to Merge.

Beginning with the first of these two points, consider the No-Tampering Condition, as described by Chomsky:

[19] Since we are assuming that Y (= (25)) *bears* the selectional list ⟨•D⟩, Y can presumably successfully be stripped of this selectional list when merging with $it_{[\text{CAT D}]}$, in keeping with clause (a) of (19).

[20] It does, however, rely on a syntactic function *cat*(x), which takes as its input a syntactic object and returns the latter's categorial feature (Merchant 2019:326). This being so, the approach is not completely free of the problems introduced by syntactic functions (on which see section 2.3 above).

(28) "One natural property of efficient computation [...] is that operations forming complex expressions should consist of no more than a rearrangement of the objects to which they apply, not modifying them internally by deletion or insertion of new elements" (Chomsky 2005:11).

(29) "Suppose X and Y are merged. Evidently, efficient computation will leave X and Y unchanged (the No-Tampering Condition NTC)" (Chomsky 2007:8).

Merchant's definition of Merge ((19)) violates the No-Tampering Condition in the way that it manipulates selectional lists. According to (19), when a selector α is merged with a selectee β, the result is a complex syntactic object $\{\gamma, \{\alpha - \ell, \beta\}\}$. As discussed above, any selectional features originally borne by α that are not satisfied/matched by β show up on γ (more precisely, on γ's selectional list). But α itself is stripped of its entire selectional list—not only the selectional feature matched by β but also any other selectional features it may originally have borne. This aspect of the definition of Merge in (19) is well motivated in the context of Merchant's (2014, 2019) system: it prevents unsatisfied selectional features from "hanging around" on α (regardless of whether their counterparts on γ or higher projections of α are eventually satisfied by other selectees or not), which is desirable, since unsatisfied selectional features are taken in Merchant's system to prevent a derivation from converging (Merchant 2014:1). But this benefit, which is ultimately empirically motivated, comes at the cost of a radical violation of the No-Tampering Condition. It is not merely that, for example, the selectional feature of α that is matched by β is, say, "struck through" ($\bullet F \rightarrow \not{\bullet F}$), which would narrowly violate the No-Tampering Condition: rather, as mentioned, α is stripped of its entire selectional list, which is to say that information originally borne by a syntactic object (α) is deleted altogether. This is, of course, a logically possible state of affairs—but insofar as "respecting the NTC is one of the fundamental, motivating ideas of the minimalist approach to grammar" (C&S:73), an assessment that strikes me as correct, we are naturally led to seek alternatives.[21]

[21] This is not to say that the No-Tampering Condition is known with certainty to constrain all syntactic operations without exception. After all, it is still a matter of considerable controversy whether late adjunction (Lebeaux 1991,

The second theoretical drawback of Merchant's definition of Merge ((19)) has to do with the set-based approach to projection it relies on. As mentioned above, (19) has the consequence that, when two syntactic objects α and β are merged, the result is a more complex syntactic object $\{\gamma, \{\alpha - \ell, \beta\}\}$, where γ is a projection of α. In other words, Merchant's definition of Merge is a descendant of the one developed in Chomsky 1995b (pp. 396ff.), which is reproduced in (30) below (though the two definitions of Merge differ in that the former, but not the latter, is explicit about the relationship between Merge and selection).

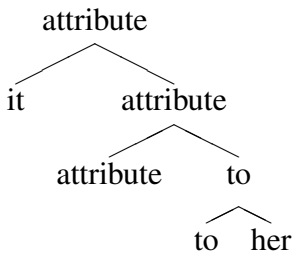
(30) Merge(α,β) = $\{\alpha, \{\alpha, \beta\}\}$

In what follows, it will be argued that the set-based approach to projection incorporated into Chomsky's definition of Merge ((30)) and inherited by Merchant's ((19)) has some theoretical drawbacks that suggest that a different approach should be pursued.

In laying out the theoretical difficulties emerging from the set-based approach to projection, it will be worthwhile to begin with a consideration that is suggestive but not conclusive. Consider again the sentence in (31) and, in particular, the structure of its VP (shown in (32) in the form of a Bare Phrase Structure–compliant tree):

(31) I would attribute it to her. (= (11))

(32)



The definition of Merge in (30) has the consequence that the tree in (32) is only an informal representation, and the "VP" in (31) is actually a set-theoretic object, namely the following one:

a.m.o.) is possible or not, even though, if it is, then adjunction can violate the No-Tampering Condition as described in (28). But whether the No-Tampering Condition constrains all syntactic operations without exception or not, violations of it should be kept to an absolute minimum, following the logic of the remark of C&S's just cited.

(33) {attribute, {it, {attribute, {attribute, {to, {to, her}}}}}}

An initial consideration—again, a suggestive rather than a conclusive one—that may lead us to suspect that the representation in (33) is on the wrong track is that, to a significant extent, it obscures syntactic relations such as sisterhood and motherhood/daughterhood that have often been taken to be linguistically significant (see, e.g., Chametzky 1995). In showing why this is so, it will be useful to introduce the notion of *setmate* (which is essentially identical to Ke's [2019] notion of co-member). A formal definition of *setmate* is given below.

(34) **Definition of *setmate***

For two distinct syntactic objects A and B, A is a *setmate* of B (alternatively: A and B are *setmates*) iff there is some syntactic object C such that A $\in$ C and B $\in$ C.

With this definition in hand, let us consider the syntactic significance of setmatehood in the representation in (33). Within {to, her} in (33), setmatehood corresponds to sisterhood in the tree in (32): in (33), *to* and *her* are setmates, and in (32), *to* and *her* are sisters. But within the "next set out" in (33)—{to, {to, her}}—setmatehood corresponds not to sisterhood in (32) but to *something like* the mother/daughter relation (i.e., immediate domination). In (33), the two members of {to, {to, her}}—namely, *to* and {to, her} (call them A and B respectively for convenience)—are setmates. B has no exact counterpart in the tree in (32), since there is no set {to, her} in this tree, but the members of B (*to* and *her*) do. A, by contrast, does at least *appear* to have an exact counterpart in (32): the *to* that is the mother of *to* and *her* (though see Chomsky 1995b:398–399 and Seely 2006:189–190 for an importantly different view, which is also discussed below). We see, then, that the setmates *to* and *her* in (33) correspond to sisters in (32)—but the setmates *to* (= A) and {to, her} (= B) in (33) do not correspond to sisters in (32); rather, the counterpart of A is (apparently) the *mother* of the counterparts of the elements of B.

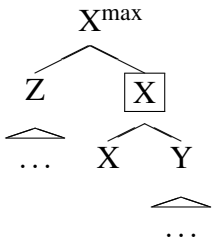
Of course, the fact that setmatehood in set-theoretic representations such as (33) does not consistently correspond either to sisterhood or to the mother/daughter relation in phrase structure trees

such as (32) does not, in and of itself, establish that the set-theoretic representations are on the wrong track. It could in principle be that sisterhood and the mother/daughter relation (as understood in the context of phrase structure trees) are not linguistically significant after all, and the set-theoretic representations are more accurate. This is why the failure of the set-theoretic representations to offer any straightforward reconstruction of sisterhood and the mother/daughter relation provides only a suggestive and not a conclusive argument against the set-based approach to projection. Closer scrutiny reveals, however, that the set-based approach to projection gives rise to some more concrete theoretical problems that suggest that this approach is indeed on the wrong track.

Consider first c-command.[22] Let us examine the set-theoretic representation in (35) and its graph-theoretic (= phrase structure tree) counterpart in (36). (In both of these representations, one syntactic element is boxed and one is annotated with a superscript "max," purely for ease of exposition. The subscript α indicates that $\alpha = \{\boxed{\text{X}}, \{\text{X}, \text{Y}\}\}$.)

(35) $\{\text{X}^{\text{max}}, \{\text{Z}, \{_{\alpha}\ \boxed{\text{X}}, \{\text{X}, \text{Y}\}\}\}\}$

(36)



Suppose we want to identify the c-command domains of various syntactic elements within (35).[23] The c-command domain of the most deeply embedded X is easily identified: it is its

[22] The issues investigated in what follows are discussed by Chomsky (1995b:398–399) and Seely (2006:189–190), the latter of whom takes them up in a context somewhat different from the present one (in particular, against the backdrop of Collins 2002). We will draw on some of Chomsky's and Seely's insights below.

[23] Being able to do this is important even if, as argued by Hornstein (2009:chap. 2), the syntactic significance of c-command is derivative (i.e., even if c-command is not even close to being a primitive of the theory). It is important, for example, for purposes of ultimately settling the question of whether structure-building features that trigger Internal Merge (under c-command) project from the lexical items that bear them to higher levels of structure (as one would expect in a Bare Phrase Structure universe—but see Ko's [2007, 2014] and Davis's [2019] arguments that they do not), and for purposes of settling the analogous question about features that trigger agreement (or Agree, if the latter exists). See Béjar and Rezac 2009, Clem 2020, and works cited therein for arguments that features triggering Agree

setmate, namely Y. This might initially lead us to expect that, for any syntactic object δ, the c-command domain of δ should be the setmate of δ. If this were so, then the c-command domain of the "next X out" in (35), namely $\boxed{\text{X}}$, would be its setmate, {X, Y}, and therefore $\boxed{\text{X}}$ would c-command the members of its setmate, namely X and Y. But as we can see from (36), this is surely the wrong result: in phrase structure tree terms, $\boxed{\text{X}}$ does not c-command X and Y but rather dominates them. The tree in (36) suggests that the actual c-command domain of $\boxed{\text{X}}$ is Z. That being so, whereas $\boxed{\text{X}}$ in the tree in (36) c-commands Z (and does not c-command X and Y, but rather dominates them), the closest we can come to reconstructing this claim in the context of the set-theoretic representation in (35) is to say that the entire syntactic object α = {X, {X, Y}} has Z as its c-command domain (cf. Chomsky 1995b:398–399, Seely 2006:189–190). $\boxed{\text{X}}$, by contrast, must be understood as not c-commanding anything in (35): $\boxed{\text{X}}$ does not c-command {X, Y}, for the reasons discussed above, and it would surely be implausible to say that $\boxed{\text{X}}$ c-commands Z, the setmate of the syntactic object α that immediately contains $\boxed{\text{X}}$.

Of course, all of this is known: the general picture emerges clearly from Chomsky 1995b (pp. 398–399) and is lucidly explained by Seely (2006:189–190). But it will be worthwhile in this context to examine some of the consequences of this general picture in greater detail.

As Seely (2006:190) notes, "[t]here is something of a mis-match between the graph-theoretic and the set-theoretic representations of [Bare Phrase Structure] phrases." If we take the tree in (36) as our starting point, the natural inference is that $\boxed{\text{X}}$ has Z as its c-command domain, but if we take the set-theoretic representation in (35) as our starting point, we are led to the view that what c-commands Z is not $\boxed{\text{X}}$ but α = {$\boxed{\text{X}}$, {X, Y}}.[24] As mentioned above, if we adopt the set-theoretic approach, we need some way of guaranteeing that $\boxed{\text{X}}$ in (35) will not c-command anything. The standard way of doing this (in the context of Bare Phrase Structure) makes use of Chomsky's (1995b:399) formal definition of *term* (where the *terms* of a syntactic object are the

do project; see Merchant 2014 (p. 3) for a formalization of the view that "no inflectional features project. Inflectional features are therefore found only on heads, never on projections." See also Abels 2003 (sect. 2.2.2.3) for arguments that certain features of heads do not project.

[24]Furthermore, as Seely points out, if we take it that the set-theoretic representation is the more accurate one (as argued in Chomsky 1995b:398–399), then the tree in (36) is but an informal representation, and to the degree that it is legitimate, what c-commands Z in (36) is not $\boxed{\text{X}}$ but "the sub-tree of which it is the root" (ibid:399).

"functioning elements" [ibid.] within it):

(37) "[F]or any structure K,

a. K is a term of K;

b. if L is a term of K, then the members of the members of L are terms of K" (Chomsky 1995b:399).

With this definition of term in mind, consider again the set-theoretic object in (35), which is repeated in (38) below, with extra annotations for convenience:

(38) $\zeta = \{X^{max}, \{_{\beta}\ Z, \{_{\alpha}\ \boxed{X}, \{_{W}\ X, Y\}\}\}\}$

According to the definition of term in (37), the terms of ζ are ζ itself; the members of the members of ζ (i.e., Z and α), and the members of their members (i.e., X and Y). That is, the terms of ζ are ζ, Z, α, X, and Y, but not X^{max}, β, $\boxed{X}$, or W.

If only the terms of a syntactic object K are the syntactically "functioning elements" of K, and if only these can enter into syntactic relations such as c-command, then, as Seely (2006) points out, we derive the desired result: that nonterms, such as $\boxed{X}$ in (38), do not c-command anything (and are themselves not c-commanded by anything). In fact, nonterms are syntactically inert quite generally.

But as pointed out by Chametzky (2000:128, 130), the definition of term in (37), which is needed to derive the result that elements such as $\boxed{X}$ in (38) do not c-command, is conceptually unnatural and thus theoretically suspect.[25] According to the definition of term in (37), the terms of a syntactic object K are K itself; the members of the members of K (call these elements κ and λ); the members of the members of κ and λ; and so on. In other words, termhood has a "skipping" property to it: the terms of K are K itself, the members of the members of K (= κ and λ), and

[25]The argument to follow is formulated in somewhat different terms than is Chametzky's (2000:128) argument against the definition of term in (37), but it is quite similar to the latter in content and very similar to it in spirit. Thus, Chametzky's argument is an important precursor (in fact, probably the most important precursor) to the argument to follow.

so on, but not the *members* of K (and not the members of κ and λ, and so on). Of course, this is logically possible; formally, there is no problem. But to gerrymander the definition of term in such a way that termhood "skips" every other level of structure is, as mentioned, conceptually unnatural; in effect, the definition of term in (37) is an artificial device—in fact, an epicycle—needed to get c-command to work out correctly (and, more generally, to guarantee that elements such as $\boxed{\text{X}}$ in (38) will be syntactically inert). It is fairly clear that the apparent need for this epicycle is a consequence of an aspect of the set-based approach to projection flagged as suspicious above: that setmatehood is being called upon to reconstruct both sisterhood and (something roughly similar to) the mother/daughter relation—i.e., immediate domination (speaking in phrase structure tree terms). Differently put, the apparent need for the epicycle strongly suggests that too much pressure is being placed on the notion of setmatehood: setmatehood is being asked to do too much. The correct minimalist response to this situation is clear: to seek out an alternative approach to projection that does not require epicycles such as the one just discussed.

Recapitulating, then, Merchant's (2019) definition of Merge compares favorably with C&S's in that it does not rely on a syntactic function Triggers and does not require the addition to the theory of a syntactic function Label, but it has two theoretical drawbacks of its own:

(39) a. It radically violates the No-Tampering Condition.

b. It incorporates (a version of) Chomsky's (1995b) set-based approach to projection, which puts too much pressure on setmatehood (relying on it to reconstruct both sisterhood and something like the mother/daughter relation), and therefore makes it difficult to reconstruct the notion of c-command without epicycles (such as a gerrymandered definition of term; cf. Chametzky 2000:128, 130).

That being so, the following section will develop two alternative formal definitions of Merge that avoid both the theoretical difficulties arising from C&S's definition (see (18)) and those arising from Merchant's definition ((39)).

4. An Alternative, Graph-Theoretic Approach to Merge

The discussion above of C&S's definition of Triggered Merge (section 2) and Merchant's (2019) definition of Merge (section 3) has a curious and perhaps unexpected implication. Both C&S and Merchant define Merge as a set-theoretic operation. According to C&S's definition, Merge(A,B) = {A, B}: Merge builds structure but does not incorporate projection (or labeling). According to Merchant's definition, Merge(A,B) = {γ, {A, B}}, where γ is a projection of the selector—i.e., Merge does incorporate projection (rather than being a more minimal structure-building operation, à la Simplest Merge).

It was argued above that a C&S-style set-theoretic definition of Merge that does not incorporate projection is problematic because it requires the postulation of freestanding syntactic functions, which leads to a number of theoretical problems ((18)). But it was also argued above that a Merchant 2019–style (or, more generally, a Chomsky 1995b–style) set-theoretic definition of Merge that does incorporate projection is also problematic, because it puts too much pressure on set-matehood, leading to the theoretical drawbacks discussed in section 3 and summarized in (39-b). But if Merge as set formation without projection is problematic, and Merge as set formation with projection is also problematic, then Merge as set formation is problematic, period.

This suggests that, in order to solve the theoretical problems arising from the definitions of Merge so far examined, we should drop the assumption that Merge involves set formation—and, more generally, the assumption that Merge is set-theoretic in nature. But if Merge does not build set-theoretic objects, then what sorts of objects does it build? A natural possibility is phrase structure trees. In what follows, it will be argued that, when a syntactic object is represented by means of a phrase structure tree, the latter is not merely a convenient informal notation for the real set-theoretic structure (contra Chomsky 1995b:397ff.; see Seely 2006:189ff. for discussion) but rather is the real structure itself. Differently put, if we define Merge as a graph- rather than as a set-theoretic operation, the resulting definition will more clearly and straightforwardly reflect the true properties of Merge.[26]

[26]Other graph-theoretic approaches to Merge and/or phrase structure are developed—in a broad range of different contexts, and on the basis of widely varying sets of assumptions—in Yasui 2003, 2004a,b, 2006, Franco 2008

The discussion will proceed as follows. Section 4.1 develops a graph-theoretic definition of Merge that respects the No-Tampering Condition,[27] and shows how it works by means of a sample derivation. Section 4.2 develops a different graph-theoretic definition of Merge that narrowly violates the No-Tampering Condition, and illustrates how it works by means of a sample derivation as well. Section 4.3 compares the two definitions, commenting on the benefits and costs of each.

There is an important issue that should be mentioned before we proceed to these definitions. In the conclusion to their paper, C&S note (p. 75): "One lesson from this exercise is that it is not possible to define Merge in isolation, independently from a network of definitions articulating the notion of a derivation. To define Merge and its recursive application properly, we had to define lexical item, lexical item token, syntactic object, a stage in a derivation, the operation Select, and the derive-by-Merge relation between stages, among other notions." This assessment seems quite correct. Nonetheless, in the interest of keeping this paper relatively targeted and focused, we will not attempt to achieve the comprehensiveness of C&S's formalization or of Merchant's (2014, 2019) formalization of minimalist syntax. Instead, we will restrict ourselves to providing formal definitions of exactly three key notions—syntactic object, Merge, and matching (the last of these being a concept that our definitions of Merge will depend on)—as well as illustrating how the relevant definitions work by means of sample derivations. We will leave for future work the important task of embedding our definitions of syntactic objecthood, Merge, and matching in a more comprehensive formalization of minimalist syntax (one that also defines *lexical item*, *categorial feature*, *derivation*, and other key concepts), thereby facilitating comparison of different approaches.

(esp. chap. 3), McKinney-Bock 2013, McKinney-Bock & Vergnaud 2013, Krivochen 2020, and references cited in those works, to which the interested reader is referred (cf. also De Vries 2009). The approach developed below differs from those developed in the works just cited in that it is more centrally concerned with the relationship between Merge and selection (including L-selection). It also differs from many of them in that it "stick[s] to [...] more mainstream [...] structures and operations" and "make[s] an effort to stay close to mainstream formulations" (to borrow two apt turns of phrase from C&S:43).

[27] *No-Tampering Condition* will occasionally be abbreviated *NTC* below.

4.1 An NTC-Obeying Graph-Theoretic Definition of Merge

4.1.1 *Defining syntactic objects, Merge, and matching*

We will begin by defining *syntactic object*:

(40) **Definition of *syntactic object***

X is a *syntactic object* iff

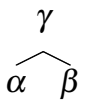
(i) X is a lexical item, or

(ii) X is a tree with root R such that every daughter of R is the root of a syntactic object.[28]

This definition of syntactic object (which is modeled after C&S's in (2) above) allows us to define Merge as follows:

(41) **Definition of Merge**

For any syntactic objects α, β, where the root of α bears a nonempty selectional list ℓ = $\langle$[•F$_1$•], …, [•F$_n$•]$\rangle$, and β matches [•F$_1$•], Merge(α,β) =



where γ is obtained from α by replacing ℓ with $\ell' = \ell$ – [•F$_1$•].

This definition of Merge (which is modeled after Merchant's [2019] definition of Merge in (19) above) relies on the notion of matching, which is defined as follows:

[28] Note that "the root of a syntactic object" can be either the root of a nontrivial tree (e.g., the root of the tree corresponding to *to her*) or simply a lexical item (e.g., *her*). Since a lexical item LI is a trivial tree (cf. Partee, ter Meulen & Wall 1990:441), LI is a syntactic object whose root is LI itself.

(42) **Definition of *match***

For any selectional feature [•F•] and syntactic object X with root R, X *matches* [•F•] iff [•F•] is keyed to

(i) the categorial feature of R, or

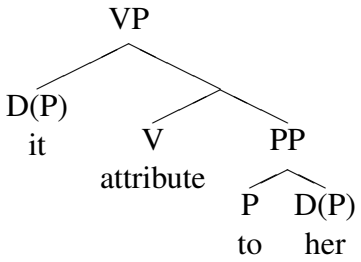
(ii) the lexical identity of R.[29]

4.1.2 *A sample derivation*

Let us see how the definition of Merge in (41) works. Consider again the sentence in (43), whose VP is shown (with informal node labels) in (44) (= (12)).

(43) I would attribute it to her. (= (11))

(44)



According to the definition of Merge in (41), this VP is built up as follows. We take the lexical items *to* and *her* to have the following lexical entries:

(45) a. $\text{to}_{[\text{CAT P, SEL } \langle [\bullet \text{D} \bullet] \rangle]}$

b. $\text{her}_{[\text{CAT D}]}$

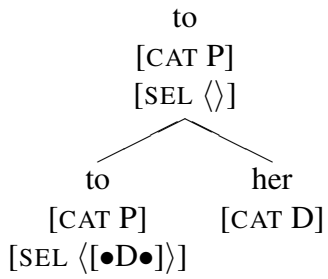
Her, like all lexical items, is a trivial tree—i.e., one consisting of only one node, namely itself. That being so, *her* can function as the X in the definition of *match* in (42): *her* is a syntactic object X with a root R (= *her* itself). Now consider the sole selectional feature borne by *to*: [•D•]. By clause (i) of the definition of *match* ((42)), the syntactic object *her* matches the selectional feature

[29] Clause (i) here covers c-selection, and clause (ii) covers L-selection. If Merchant (2019) and Hewett (2020) are right to unify L-selection and c-selection (by positing that, e.g., the categorial feature of *to* is [CAT to], not [CAT P]; see note 17), then the disjunction in this definition can be eliminated in favor of what is currently its first disjunct.

[•D•] on *to* because the root of this (trivial) tree—namely, *her* itself—bears a categorial feature ([CAT D]) that that selectional feature is keyed to.

Since the syntactic object *to* bears a nonempty selectional list, and the syntactic object *her* matches the first selectional feature on that list, the definition of Merge in (41) has the consequence that Merge(to, her) is defined (with *to* being α, the selector, and *her* being β, the selectee). That operation outputs a tree consisting of a root γ with two daughters, α and β. The element γ is identical to α except that its selectional list is not ℓ (the selectional list of α) but rather ℓ', which is obtained from ℓ by subtracting the latter's first selectional feature. The output of Merge(to, her)—call it μ for convenience—is shown below:

(46)

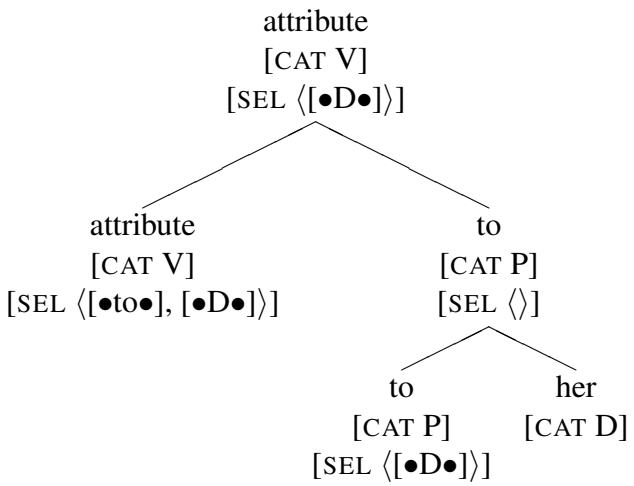


Continuing the derivation, we take the V *attribute* to have the following lexical entry:

(47) attribute$_{[\text{CAT V, SEL } \langle [\bullet\text{to}\bullet], [\bullet\text{D}\bullet] \rangle]}$

The root ρ of μ (= (46)) is *to*$_{[\text{CAT P, SEL } \langle\rangle]}$. The first selectional feature on *attribute*'s selectional list, [•to•], is keyed to the lexical identity of ρ: [•to•] requires that its bearer merge with a syntactic object whose root is a *to*-type element (i.e., one that is, or is projected from, the lexical item *to*), not merely, say, an element bearing the categorial feature [CAT P]. That being so, μ matches [•to•] on *attribute* (by clause (ii) of the definition of *match* in (42)), and the definition of Merge in (41) has the consequence that Merge(attribute$_{[\text{CAT V, SEL } \langle [\bullet\text{to}\bullet], [\bullet\text{D}\bullet] \rangle]}$, μ) is defined. This operation yields the following tree (call it π):

(48)

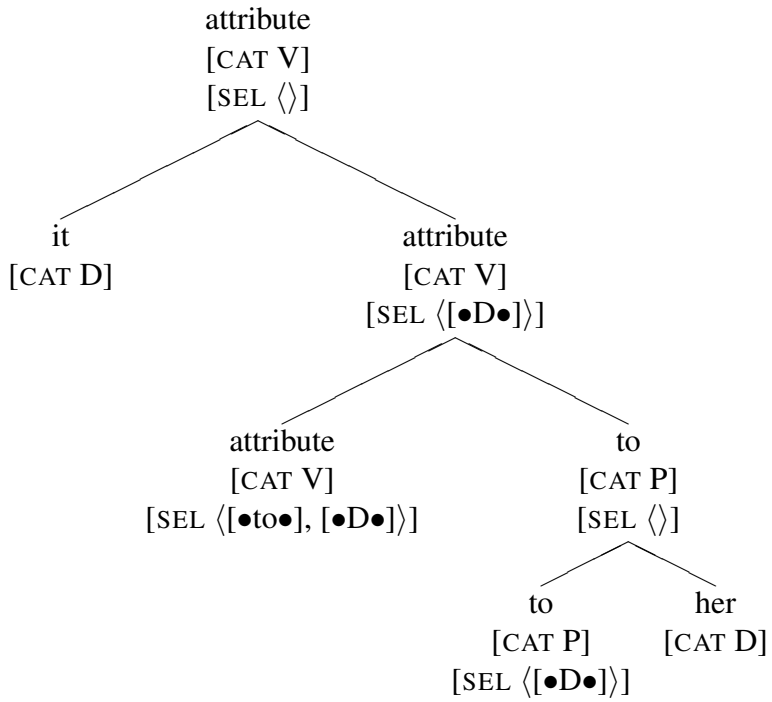


Finally, we take the lexical item *it* to have the following lexical entry:

(49) $\text{it}_{[\text{CAT D}]}$

Since the syntactic object *it* (= (49)) matches [•D•] on the root of π (= (48)), Merge(π, $\text{it}_{[\text{CAT D}]}$) is defined, and outputs the following tree:

(50)



This concludes the derivation of the VP in (44).

4.1.3 *Evaluating the NTC-obeying definition of Merge in (41)*

Now that we have seen the version of Merge defined in (41) in action in a derivation, we can proceed to evaluate it on theoretical grounds.

Like C&S's definition of Triggered Merge ((6)), but unlike Merchant's (2019) definition of Merge ((19)), the definition of Merge in (41) obeys the No-Tampering Condition. According to (41), when a syntactic object α merges with a syntactic object β that matches α's first selectional feature, that selectional feature is never actually deleted, or even checked in a way that would alter α (e.g., by being "struck through," as in $[\bullet F_1 \bullet] \rightarrow [\cancel{\bullet F_1 \bullet}]$). Rather, that selectional feature simply does not appear on the selectional list of γ, the mother of α and β.[30] This approach to "feature satisfaction" is modeled closely after C&S's, which they incorporate into their system by means of their Triggers function ((5)). That the definition of Merge in (41) obeys the No-Tampering Condition is, of course, a significant point in its favor, following the logic of the discussion in section 3.2.

There are, however, some theoretical worries one might have about the definition of Merge in (41). Consider the way that (41) implements the NTC-compliant approach to feature satisfaction just discussed: by means of the clause "where γ is obtained from α by replacing ℓ with $\ell' = \ell - [\bullet F_1 \bullet]$." First, since this clause of (41) *replaces* the selectional list ℓ with a different selectional list ℓ', it arguably misses to some degree the generalization that ℓ and ℓ' are very similar (the latter being identical to the former except that it is missing the former's first selectional feature). Alternatively, one can look at this theoretical issue in a different way. A selectional feature (e.g., $[\bullet F_1 \bullet]$) is a "problem" for the system,[31] in that it requires that its bearer merge with a syntactic object that matches it, on pain of a derivational crash—whereas, if the selectional feature simply were not there, this pressure would not be placed on the system. Given that, and given that the

[30] On this approach, then, the mere presence of an unsatisfied/unchecked selectional feature on a nonroot does not cause the derivation to crash. But because there is abundant empirical evidence that selectional features must be satisfied (even if feature "satisfaction" is cashed out in an NTC-obeying way, as it is here), it will be necessary on this approach to define convergence in such a way that a derivation that ultimately yields a syntactic object X can converge only if the root of X bears no selectional features. The task of defining *derivation*, *convergence*, and related concepts in the context of this approach must be left for future work.

[31] A virus, to use the term that Lasnik (1999) employs in discussing a proposal made by Chomsky (1995a:chap. 4). (Lasnik notes [p. 199] that the term *virus* was suggested to him by Juan Uriagereka.)

definition of Merge in (41) replaces ℓ with something (in constructing γ), it is not clear a priori why it does not simply replace ℓ with the empty list $\langle\rangle$, thereby avoiding creating further problems for the system. Yet empirically, it is clear that it does not do this, because if it did, we would get, e.g., **I would attribute to her*, since there would be no need to merge a D(P) such as *it* with any projection of *attribute*.

Another theoretical issue raised by the definition of Merge in (41) is the following. As mentioned above, this definition includes the clause "where γ is obtained from α by replacing ℓ with $\ell' = \ell - [\bullet F_1 \bullet]$." Although this obeys the letter of the No-Tampering Condition (see (28) and (29)), because it leaves α itself (along with β) unchanged, it arguably violates its spirit. This is because the procedure for constructing γ that is written into this definition of Merge is not very different from "cloning" the root of α and then "reaching into" the clone and deleting the first feature on its selectional list—i.e., in effect, violating the integrity[32] of a lexical item, even if a nonterminal node is not technically a lexical item.

In view of these theoretical issues, it will be worthwhile to consider another possible graph-theoretic definition of Merge.

4.2 A Narrowly NTC-Violating Graph-Theoretic Definition of Merge

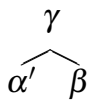
4.2.1 *Defining Merge and matching*

We will begin by defining Merge as follows:

[32]The concept being invoked here should not be confused with "lexical integrity" in the sense argued for by proponents of the Lexicalist Hypothesis (e.g., Bresnan & Mchombo 1995; see their n. 1 for many more references). Nothing in the proposals made here requires a commitment to the correctness (or, for that matter, to the incorrectness) of the Lexicalist Hypothesis.

(51) **Definition of Merge**

For any syntactic objects[33] α, β, where the root of α bears a nonempty selectional list ℓ = $\langle \ldots, [\bullet F_1 \bullet], \ldots, [\bullet F_n \bullet] \rangle$, where $[\bullet F_1 \bullet]$ is the first unsatisfied feature on ℓ, and β matches $[\bullet F_1 \bullet]$, Merge(α,β) =



where α' is obtained from α by replacing $[\bullet F_1 \bullet]$ with ~~$[\bullet F_1 \bullet]$~~ (i.e., $[\bullet F_1 \bullet]$, but satisfied), and γ is a reflection of α'.

(*Reflection* is defined in (53) below.) As in section 4.1.1, *match* is defined as follows:

(52) **Definition of *match*** (= (42))

For any selectional feature [•F•] and syntactic object X with root R, X *matches* [•F•] iff [•F•] is keyed to

(i) the categorial feature of R, or

(ii) the lexical identity of R.

Unlike the definition of Merge in (41), the definition of Merge in (51) relies on a notion of *reflection*, which is defined as follows:

(53) **Definition of *reflection***

For two nodes A and B, B is a *reflection* of A iff B and A are identical at every stage of the derivation at which they both exist (i.e., whatever happens to B also happens to A, and vice versa).

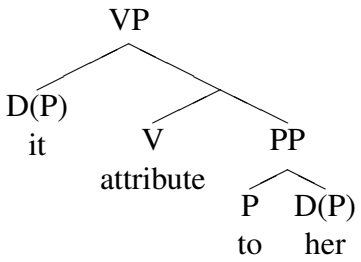
To see how these definitions work, let us consider a sample derivation.

[33] *Syntactic object* is still defined as in (40) above.

4.2.2 *A sample derivation*

Consider once more the VP shown in (54) (= (12)):

(54)



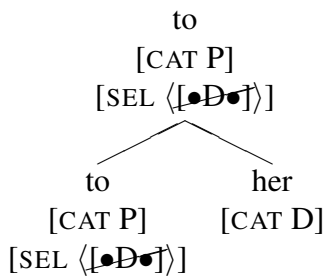
As in section 4.1.1, we take the lexical items *to* and *her* to have the following lexical entries:

(55) a. $\text{to}_{[\text{CAT P, SEL } \langle [\bullet \text{D} \bullet] \rangle]}$

b. $\text{her}_{[\text{CAT D}]}$

Since the syntactic object $\textit{her}_{[\text{CAT D}]}$ matches [•D•] on *to*, the definition of Merge in (51) has the consequence that Merge(to, her) is defined, with *to* = α (the selector) and *her* = β (the selectee). When *to* is merged with *her*, *to* is replaced by a version of itself (α') whose first (and only) selectional feature, [•D•], is satisfied ("struck through"): [~~•D•~~]. Furthermore, γ, the newly created mother of α' and β, is a reflection of α'—i.e., it is identical to α' from this point forward. As a result, Merge(to, her) yields the following tree (call it σ for convenience):

(56)



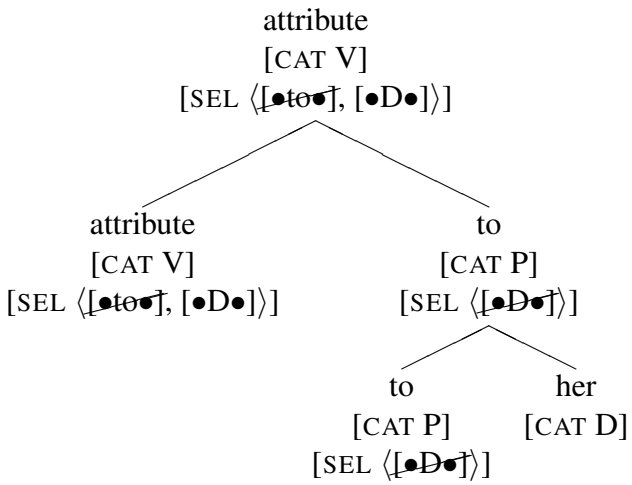
Continuing the derivation, we take the V *attribute* to have the following lexical entry:

(57) $\text{attribute}_{[\text{CAT V, SEL } \langle [\bullet \text{to} \bullet], [\bullet \text{D} \bullet] \rangle]}$ (= (47))

Since σ (= (56)) matches the first selectional feature of *attribute*, namely [•to•], Merge(attribute,

σ) is defined. When *attribute* and σ are merged, the first selectional feature of *attribute* is satisfied (struck through)—and so is the first selectional feature of the mother of the lexical item *attribute*, which is a reflection of the latter. The output of Merge(attribute, σ)—call it τ—is shown below:

(58)

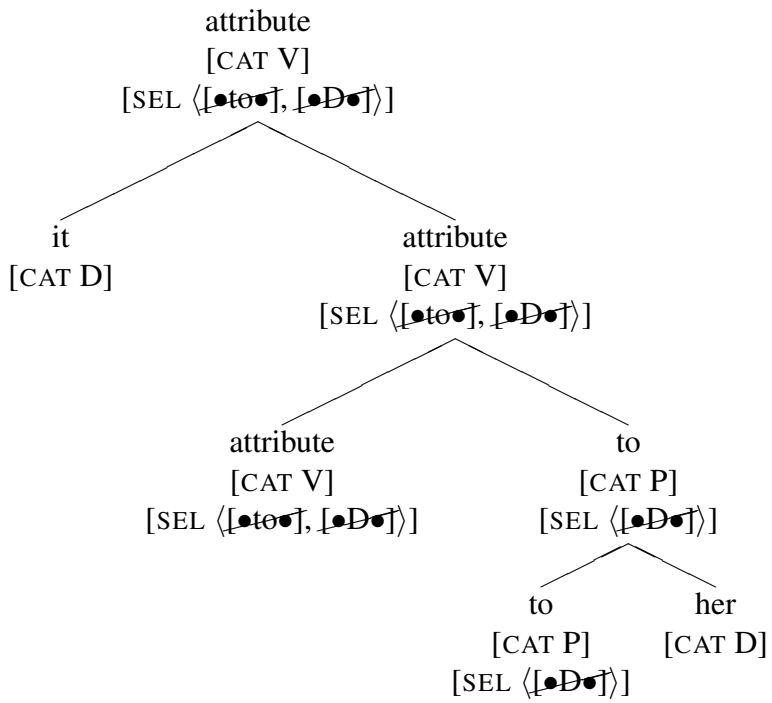


Finally, we take *it* to have the following lexical entry:

(59) $\text{it}_{[\text{CAT D}]}$ (= (49))

Since the syntactic object *it* (= (59)) matches [•D•] on the root of τ (= (58)), Merge(τ, it) is defined, and yields the following tree:

(60)



Crucially, Merge(τ, it) causes [•D•] to be satisfied (struck through, becoming [•~~D~~•]) not only on the root of τ, and not only on the mother of the root of τ (which is a reflection of the root of τ), but also on the *attribute* node that is a daughter of the root of τ. Since Merge(τ, it) satisfies [•D•] on the root of τ, and the root of τ is a reflection of its *attribute* daughter (call the latter χ), the Merge operation in question also satisfies [•D•] on χ.[34] This is because reflectionhood is symmetric ((53)): if a node A is a reflection of its daughter B, then B is also a reflection of A (its mother). This concludes our sample derivation.

4.3 Comparing the Two Graph-Theoretic Definitions of Merge

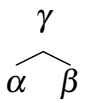
Before concluding, it will be worthwhile to compare the two graph-theoretic definitions of Merge just developed, highlighting the benefits and costs of each. The NTC-obeying graph-theoretic definition of Merge is repeated in (61) below, and the narrowly NTC-violating graph-theoretic

[34] If Merge(τ, it) did not apply, [•D•] would remain unsatisfied both on the root of τ and on its *attribute* daughter (χ). On this approach to Merge (but not on the NTC-obeying approach developed in section 4.1), it is natural to posit that a syntactic object containing a node bearing an unsatisfied feature cannot undergo Transfer, so satisfaction of selectional features (by applications of Merge) is indirectly forced. The reflectionhood component of the definition of Merge in (51) has the beneficial consequence that, for a syntactic object X with root R that contains some nonroot node Y, an unsatisfied feature on Y can (still) be satisfied as long as R is a reflection of Y. We will leave for future work the task of defining Transfer in the context of this approach.

definition of Merge is repeated in (62).

(61) **Graph-theoretic definition of Merge (NTC-obeying)** (= (41))

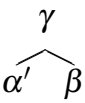
For any syntactic objects α, β, where the root of α bears a nonempty selectional list ℓ = $\langle$[•F$_1$•], …, [•F$_n$•]$\rangle$, and β matches [•F$_1$•], Merge(α,β) =



where γ is obtained from α by replacing ℓ with ℓ' = ℓ – [•F$_1$•].

(62) **Graph-theoretic definition of Merge (narrowly NTC-violating)** (= (51))

For any syntactic objects α, β, where the root of α bears a nonempty selectional list ℓ = $\langle$…, [•F$_1$•], …, [•F$_n$•]$\rangle$, where [•F$_1$•] is the first unsatisfied feature on ℓ, and β matches [•F$_1$•], Merge(α,β) =



where α' is obtained from α by replacing [•F$_1$•] with ~~[•F$_1$•]~~ (i.e., [•F$_1$•], but satisfied), and γ is a reflection of α'.

The main difference between (61) and (62) has to do with what happens when a syntactic object α whose root bears a selectional feature [•F$_1$•] merges with a syntactic object β that matches [•F$_1$•]. According to (61), [•F$_1$•] on α is not deleted or even checked ("struck through") as such; instead, it simply does not show up on the selectional list of γ, the mother of α and β. (By contrast, the other selectional features of α are mirrored by identical features on the selectional list of γ.) According to (62), on the other hand, what happens is that α is replaced by a version of itself (α') whose [•F$_1$•] feature is satisfied (struck through)—and because γ (the mother of α') is a reflection

of α', [•F_1•] on γ is satisfied too.

With that established, we can return to the potential theoretical drawbacks of (61) discussed in section 4.1.3, determine to what degree (62) overcomes them, and discuss any drawbacks that (62) itself may have.

The first potential drawback of (61) mentioned in section 4.1.3 had to do with the clause "where γ is obtained from α by replacing ℓ with $\ell' = \ell$ – [•F_1•]." As mentioned there, since this clause *replaces* ℓ with a different selectional list ℓ', it arguably misses to some extent the generalization that ℓ and ℓ' are very similar, since it could just as easily have replaced ℓ with something else. In particular (and looking at the issue from a partially different perspective), since a selectional feature is a problem for the system—a virus in Lasnik's (1999) sense; see note 31—it is theoretically unclear why, in constructing γ, (61) does not simply replace ℓ with the empty list $\langle\rangle$, thereby avoiding saddling the system with further problems. But empirically, it must not, since the Merge operation triggered by α's first selectional feature clearly does not obviate the need for subsequent Merge operations triggered by α's other selectional features (or, more precisely, by their counterparts on projections of α). In summary, (61) states that Merge(α,β) constructs an entirely new element γ; therefore, given (61), it is theoretically unclear why the system does not capitalize on this state of affairs to ensure that γ will be free of selectional features, thereby avoiding future problems (the need to satisfy yet more selectional features originally introduced by α).

By contrast, the narrowly NTC-violating definition of Merge in (62) arguably does better on this score. This claim might initially seem surprising, because (62) contains a clause "where α' is obtained from α by replacing [•F_1•] with [•~~F_1~~•] (i.e., [•F_1•], but satisfied)." A skeptic might ask about (62) a version of the question asked above about (61): why the relevant clause of (62) does not read "where α' is obtained from α by replacing every unsatisfied feature [•F_k•] with [•~~F_k~~•] (i.e., the same feature, but satisfied)."

Two points are worth making in response. First, simultaneously satisfying (or checking, or "striking through") multiple features at once is a more complex (sub)operation than satisfying only one, so it would be unsurprising, from a minimalist standpoint, for the latter (sub)operation

to be favored. By contrast, since (61) obtains γ from α by replacing ℓ wholesale with something—ℓ'—it is far from obvious that it is simpler for that something to be a specially constructed list ℓ – [•F$_1$•] than for it to simply be the empty list $\langle\rangle$.[35] Secondly, (62) captures in a fairly direct way the intuition that the reason [•F$_1$•] is satisfied, but not the other selectional features of α, is that β matches the former but not (necessarily) the latter. That is, [•F$_1$•] is the only feature of α that has been verified to be a match for something (in particular, β). This intuition is captured less directly by (61). According to (61), too, [•F$_1$•], but not the other selectional features of α, has been verified to be a match for β, and this is (indirectly) linked to the fact that only [•F$_1$•] is subtracted from ℓ in forming ℓ'. But because the formation of ℓ' is part of the process of creating γ, an element distinct from α and β, there is a certain amount of "backtracking" or reference to a "prior" operation involved in the creation of γ. That is, information about the interaction between β and a feature of α must be "remembered" in order to form the distinct element γ.[36] According to (62), by contrast, the construction of γ is trivial. The element γ is simply a reflection of α, so it is perfectly clear why γ bears a satisfied selectional feature ~~[•F$_1$•]~~ but also, in some cases, unsatisfied selectional features (= viruses) that the system still has to deal with: Merge(α,β) satisfies only one of α's selectional features and leaves the rest unsatisfied, and whatever happens to (or on) α happens to (or on) its reflection, namely γ.

The other possible theoretical drawback of (61) mentioned in section 4.1.3 also had to do with the clause "where γ is obtained from α by replacing ℓ with $\ell' = \ell$ – [•F$_1$•]." As noted there, although this obeys the letter of the No-Tampering Condition (because α itself, along with β, remains unaltered) it arguably violates its spirit: the procedure described in that clause is not very different from "cloning" the root of α and then "reaching into" the clone (call it γ) and

[35] Such arguments must, however, be made with caution in the absence of an explicit complexity metric, as alluded to by Epstein, Groat, Kitahara, and Kawashima (1998:18); Epstein (1999:319); and Seely (2006:n. 6).

[36] It might be objected that the use of the temporally or procedurally oriented terms "backtracking," "prior," and "remembered" is inappropriate in this connection, even in the context of the derivational framework assumed here, because, according to (61) as actually formalized, Merge is a single operation whose suboperations are simultaneous rather than derivationally ordered. In my assessment, this would be a technical objection that would not truly defuse the conceptual argument made in the text. But the skeptic can simply reformulate the argument by cashing it out not in terms of (technically veiled, implicit) derivational timing but rather in terms of relative nonlocality: information about an interaction between β and a feature of α is crucially involved in constructing γ, an entirely distinct element—a complication in the definition of Merge that should be avoided if possible.

deleting the first feature on γ's selectional list altogether. This raises the question of why a system powerful enough to do the equivalent of that would not also be able to delete features of α itself (thereby violating not only the spirit but also the letter of the No-Tampering Condition). How, then, does the definition of Merge in (62) compare to that in (61) in this respect? The definition in (61) arguably violates the spirit of the No-Tampering Condition, but that in (62) violates its letter—though narrowly, by striking features through (i.e., by implementing feature satisfaction in a way that alters α), not radically, by deleting them altogether, the way that Merchant's (2019) definition of Merge does. At any rate, this consideration would initially seem to favor (61) over (62), following the logic of the discussion in section 3.2. But the issue is not so clear-cut: if not only the (narrowly) NTC-violating definition of Merge but also the NTC-obeying one violates the spirit of the No-Tampering Condition, then it is not clear that the latter is *significantly* superior to the former on this theoretical criterion.

Summing up, then, the two graph-theoretic definitions of Merge laid out in this section each have their benefits and costs. The definition in (61) has the advantage that it obeys the No-Tampering Condition, but at the cost of partially missing the generalization that ℓ' (the selectional list of γ) is very similar to ℓ (the selectional list of α)—and making it unclear why, in constructing γ from α, ℓ is replaced by $\ell' = \ell - [\bullet F_1 \bullet]$ rather than simply by the empty list $\langle\rangle$, the latter of which would leave the system with fewer problems to solve (i.e., fewer selectional features or "viruses" to deal with by subsequent applications of Merge). The definition in (62), by contrast, makes it clear why, following Merge(α,β), the unsatisfied features of α' (the modified version of α) are also present on γ: because β satisfies only the selectional feature of α that it matches, and γ is a reflection of α', so γ and α' are identical at every stage of the derivation at which they both exist. The cost of this theoretical success, though, is that (62) violates the No-Tampering Condition—though narrowly, by striking features through (adding information to α), rather than radically, by deleting them altogether (subtracting information from α).

5. Conclusion

This paper has investigated the properties of the fundamental syntactic structure-building operation Merge, taking as its starting point two recent formal definitions of this operation: Collins and Stabler's (2016) definition of Triggered Merge and Merchant's (2019) definition of Merge. Although these definitions have significant strengths—among them that they are both explicit about the close relationship between Merge and selection—they also have some theoretical drawbacks. More broadly, the paper has argued that set-theoretic definitions of Merge in general (not just the two examined here) run into conceptual problems—regardless, crucially, of whether they incorporate projection as part of Merge or not.

This intermediate conclusion motivated the proposal that Merge is not set-theoretic but graph-theoretic in nature—and, more specifically, that the syntactic objects that Merge operates on and builds are (Bare Phrase Structure–compliant) trees, not sets (see note 26 above for relevant references). Two graph-theoretic definitions of Merge, repeated most recently in (61) and (62), were proposed, and their benefits and costs were compared. The definition in (61) obeys the No-Tampering Condition, but at the cost of making it theoretically unclear why Merge(α,β), where α is the selector, satisfies only one selectional feature of α, not all of them.[37] The definition in (62), by contrast, makes that observation fall into place naturally and straightforwardly, but at the cost of violating the No-Tampering Condition—though narrowly rather than radically: a selectional feature on α, when satisfied, is "struck through" (adding information to α), but not deleted altogether (subtracting information from α).

Whichever of these two definitions or whatever modification of one of them ultimately turns out to be correct, the larger picture that emerges from this investigation is one in which Merge is best understood as a graph-theoretic rather than as a set-theoretic operation. This conclusion in turn raises the question of how a host of other key concepts in minimalist syntax (lexical item, derivation, derivational stage, categorial feature, and many others) should be understood and formally defined in this context—or, differently put, what light the graph-theoretic conception of Merge

[37] In addition, although (61) obeys the letter of the No-Tampering Condition, it arguably violates its spirit (see section 4.3).

defended here can shed on the atoms and on the other fundamental operations of syntax.

References

Abels, K. 2003. Successive cyclicity, anti-locality, and adposition stranding. Ph.D. dissertation, University of Connecticut.

Béjar, S. & M. Rezac. 2009. Cyclic Agree. *Linguistic Inquiry* 40:35–73.

Bošković, Ž. 2002. A-movement and the EPP. *Syntax* 5:167–218.

Bresnan, J. & S. Mchombo. The lexical integrity principle: Evidence from Bantu. *Natural Language and Linguistic Theory* 13:181–254.

Chametzky, R. 1994. Chomsky-adjunction. *Lingua* 93:245–264.

Chametzky, R. 1995. Dominance, precedence, and parameterization. *Lingua* 96:163–178.

Chametzky, R. 1996. *A theory of phrase markers and the extended base*. Albany, NY: SUNY Press.

Chametzky, R. 2000. *Phrase structure: From GB to Minimalism*. Malden, MA: Blackwell.

Chametzky, R. 2008 [2003]. Phrase structure. In *Minimalist syntax*, ed. Randall Hendrick, 192–226. Oxford: Blackwell.

Chametzky, R. 2011. No derivation without representation. In *The Oxford handbook of linguistic minimalism*, ed. C. Boeckx, 311–326. Oxford: Oxford University Press.

Chomsky, N. 1995a. *The minimalist program*. Cambridge, MA: MIT Press.

Chomsky, N. 1995b. Bare Phrase Structure. In *Evolution and revolution in linguistic theory: Studies in honor of Carlos P. Otero*, ed. H. Campos & P. Kempchinsky, 51–109. Washington, D.C.: Georgetown University Press.

Chomsky, N. 2005. Three factors in language design. *Linguistic Inquiry* 36:1–22.

Chomsky, N. 2007. Approaching UG from below. In *Interfaces + recursion = language?*, ed. U. Sauerland & H.-M. Gärtner, 1–29. Berlin: Mouton de Gruyter.

Clem, E. Cyclic expansion in Agree: Maximal projections as probes. Ms., UC San Diego.

Collins, C. 1997. *Local economy*. Cambridge, MA: MIT Press.

Collins, C. 2002. Eliminating labels. In *Derivation and explanation in the Minimalist Program*, ed. S.D. Epstein & T.D. Seely, 42–64. Oxford: Blackwell.

Collins, C. 2017a. Merge(X,Y) = {X,Y}. In *Labels and roots*, ed. L. Bauke & A. Blümel. Berlin: De Gruyter.

Collins, C. 2017b. A smuggling approach to the dative alternation. Ms., NYU.

Collins, C. & E. Stabler. 2016. A formalization of minimalist syntax. *Syntax* 19:43–78.

Davis, C. 2019. Possessor extraction in colloquial English: Evidence for successive-cyclicity and Cyclic Linearization. *Linguistic Inquiry*.

Epstein, S.D. 1999. Un-principled syntax: The derivation of syntactic relations. In *Working minimalism*, ed. S.D. Epstein & N. Hornstein, 317–345. Cambridge, MA: MIT Press.

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

Epstein, S.D., H. Kitahara & T.D. Seely. 2014. Labeling by minimal search: Implications for successive-cyclic A-movement and the conception of the postulate "phase." *Linguistic Inquiry* 45:463–481.

Epstein, S.D., H. Kitahara & T.D. Seely. 2015. From *Aspects*' 'daughterless mothers' (aka delta nodes) to *POP*'s 'motherless sets' (aka non-projection): A selective history of the evolution of Simplest Merge. In *50 Years Later: Reflections on Chomsky's Aspects*, ed. Á. Gallego & D. Ott, 99–112. Cambridge, MA: MITWPL.

Franco, L. 2008. Graph theory and Universal Grammar. Ph.D. dissertation, Università di Firenze.

Freidin, R. 2016. Chomsky's linguistics: The goals of the generative enterprise. Review of *Chomsky's linguistics*, ed. P. Graff & C. van Urk. *Language* 92:671–723.

Fukui, N. 2011. Merge and bare phrase structure. In *The Oxford handbook of linguistic minimalism*, ed. C. Boeckx, 73–95. Oxford: Oxford University Press.

Harley, H. 2008. The bipartite structure of verbs cross-linguistically (or: Why Mary can't *exhibit John her paintings*). In *Conferências do V Congresso Internacional da Associação Brasileira de Linguística*, ed. T. Cristófaro Silva & H. Mello, 45–84. Belo Horizonte, Brazil: ABRALIN & FALE/UFMG.

Heck, F. & G. Müller. 2007. Extremely local optimization. In *Proceedings of WECOL 26*, ed. E. Bainbridge & B. Agbayani, 170–183. Fresno, CA: California State University.

Hewett, M. 2020. Lexically selected PPs can vary by template in Semitic. Ms., University of Chicago.

Hornstein, N. & J. Nunes. 2008. Adjunction, labeling, and Bare Phrase Structure. *Biolinguistics* 2:57–86.

Hornstein, N. 2009. *A theory of syntax: Minimal operations and Universal Grammar*. Cambridge, UK: Cambridge University Press.

Ke, Hezao. 2019. The syntax, semantics and processing of agreement and binding grammatical illusions. Ph.D. dissertation, University of Michigan.

Keyser, S. J. & T. Roeper. 1992. Re: The Abstract Clitic Hypothesis. *Linguistic Inquiry* 23:89–125.

Ko, H. 2007. Asymmetries in scrambling and Cyclic Linearization. *Linguistic Inquiry* 38:49–83.

Ko, H. 2014. *Edges in syntax: Scrambling and cyclic linearization*. Oxford: Oxford University Press.

Koopman, H. 2020. On the syntax of the *can't seem* construction in English. In *Smuggling in syntax*, ed. A. Belletti & C. Collins, 188–221. Oxford: Oxford University Press.

Krivochen, D.G. 2020. Syntax as graph theory. Ms., CINN, University of Reading.

Larson, B. 2014. Russian comitatives and the ambiguity of adjunction. *Journal of Slavic Linguistics* 22:11–49.

Larson, R.K. 1988. On the double object construction. *Linguistic Inquiry* 19:335–391.

Larson, R.K. 1990. Double objects revisited: Reply to Jackendoff. *Linguistic Inquiry* 21:589–632.

Lasnik, H. 1999. On feature strength: Three minimalist approaches to overt movement. *Linguistic Inquiry* 30:197–217.

Lasnik, H. & Z. Stone. 2020. Rethinking phrase structure. In *Syntactic architecture and its consequences II: Between syntax and morphology* (Open Generative Syntax 10), ed. A. Bárány, T. Biberauer, J. Douglas & S. Vikner, 191–206. Language Science Press.

Lebeaux, D. 1991. Relative clauses, licensing, and the nature of the derivation. *Syntax and Semantics* 25:209–229.

McCloskey, J. 2000. Quantifier float and *wh*-movement in an Irish English. *Linguistic Inquiry* 31:57–84.

McKinney-Bock, K.S. 2013. Building phrase structure from items and contexts. Ph.D. dissertation, USC.

McKinney-Bock, K. & J.-R. Vergnaud. 2013. Grafts and beyond: Graph-theoretic syntax. In *Primitive elements of grammatical theory: Papers by Jean-Roger Vergnaud and his collaborators*, ed. K. McKinney-Bock & M.L. Zubizarreta, 207–236. London: Routledge.

Merchant, J. 2014. Some definitions. Ms., University of Chicago.

Merchant, J. 2019. Roots don't select, categorial heads do: Lexical-selection of PPs may vary by category. *The Linguistic Review* 36:325–341.

Partee, B., A. ter Meulen, & R.E. Wall. 1990. *Mathematical methods in linguistics*. Dordrecht: Kluwer.

Pesetsky, D. 1991. Zero syntax: Vol. 2: Infinitives. Ms., MIT, Cambridge, MA.

Punske, J. 2013. Three forms of English verb particle constructions. *Lingua* 135:155–170.

Rauber, R. 2019. Adjuncts attach differently. Ms., University of Chicago.

Seely, T.D. 2006. Merge, derivational c-command, and subcategorization in a label-free syntax. In *Minimalist essays*, ed. C. Boeckx, 182–217. Amsterdam: John Benjamins.

de Vries, Mark. 2009. On multidominance and linearization. *Biolinguistics* 3:344–403.

Yasui, M. 2002. Crash-proofness of the lexicon and Bare Phrase Structure. Ms., Dokkyo University.

Yasui, M. 2003. A graph-theoretic reanalysis of Bare Phrase Structure theory and its implications on parametric variation. Paper presented at Linguistics and Phonetics 2002, Meikai University, Urayasu, September.

Yasui, M. 2004a. Asymmetries in word order and syntactic structure as lexical graph. Ms., Dokkyo University.

Yasui, M. 2004b. Syntactic structure without projection labels. Ms., Dokkyo University.

Yasui, M. 2006. An order-free representation of syntactic structure and the head-parameter. Ms., Dokkyo University.

Zyman, E. 2020. Morphology as Syntax: A case study of the syntactic autonomy of prefixes. Talk presented at Syntactic Approaches to Morphology, NYU, December (virtual).