A formalization of Agree as a derivational operation

December 20, 2022

Abstract

Using the framework laid out by Collins and Stabler (2016), I develop a formal definition of Agree as a syntactic operation. I begin by constructing a formal definition of a version of long-distance Agree in which a structurally higher element values a feature on a structurally lower element, and modify that definition to reflect various versions of Agree that have been proposed in the "minimalist" literature. I then discuss the theoretical implications of these formal definitions, arguing that Agree requires a new conception of the lexicon, and unjustifiably violates NTC in all its non-vacuous forms.

Keywords: theory, formalization, minimalism, agree, derivations

1 Introduction

Minimalist Principles & Parameters theories of grammar deal mainly in procedures which generate linguistic expressions from atoms in an incremental fashion. That is, these theories traffic in computational procedures, in the sense of Turing, Church, Post, *et al.*,[1] that relate stage n of a derivation to stage $n+1$

---

[1] This has been true of generative grammars since their inception.Chomsky's early major publications, for instance, are replete with references to "technical devices for expressing a system of recursive processes"(1965, p8) stemming from then-recent "work in logic and foundations of mathematics" (1957, p22), which contemporary reader would have no doubt understood as reference to the work of these authors. See also Harris (2002) and Pullum (2011) for discussion of the links between generative linguistics and the work of these scholars.

of that same derivation in a regular well-defined way. From this perspective, Merge is the crown jewel of these theories. It has been developed with the two main goals of formal explicitness and descriptive adequacy. Much of the current literature in minimalist P&P grammar, however, assumes the existence of a second core procedure, Agree.

As its name suggests, Agree is the operation that causes grammatical agreement—subject-predicate agreement, case marking, etc.—which, I argue in this paper, has yet to be sufficiently defined in such a way as to properly analyze its theoretical and empirical properties.[2] The correct characterization of Agree, as with theoretical proposal, ultimately depends on empirical and theoretical considerations. Virtually the entire contemporary Agree literature, however, focuses on empirical concerns to the exclusion of theoretical questions.[3] This paper seeks to remedy this gap somewhat. The assertion that the Agree literature is primarily focused on empirical concerns to the exclusion of theoretical ones, seems to be contradicted by the sheer number of theories of Agree that have been proposed—Chomsky (2000) begins with what might be called Classical Agree, and scholars later propose Cyclic Agree (Béjar & Rezac, 2009), Local Agree (Hornstein, 2009), Fallible Agree (Preminger, 2014), and Upward Agree (Bjorkman & Zeijlstra, 2014; Zeijlstra, 2012), just to name those theories of Agree which have names. In fact, the proliferation of such theories is to be expected when inquiry is guided by the empirical rather than the theoretical, just as the proliferation of empirical predictions is to be expected when inquiry is guided by the theoretical.

Take, for instance, the recent debate regarding Upward vs Downward Agree (Preminger, 2013; Zeijlstra, 2012). This debate turns entirely on whether one version of Agree can capture a certain set of data while the other cannot. The debate tacitly assumes that both versions are definable given shared theoretical assumptions, and makes no real effort to investigate what if any implications either might have for the broader grammatical theory in which it is embedded. Indeed, the contrast between the two types of Agree seems to be an unquestioned theoretical assumption, which perhaps need not be made.

This lack of theoretical assesment of Agree is troubling, since the operation has been implicated in

---

[2] Ermolaeva (2018) defines Agree in the framework of Minimalist Grammar (MG) (Stabler, 1997). This framework, despite its name and as Collins and Stabler (2016) argue, is only tangentially related to minimalist theory and has substantially different goals and concerns. I set Ermolaeva's work aside for this reason. I also set aside alternatives to Agree embedded in other theoretical frameworks for the same reason.

[3] See Chametzky (1996) for discussion on the distinction between theoretical work and empirical work.

a wide range of grammatical phenomena beyond the morphological agreement phenomena from which it gets its name. Agree has been argued to be necessary to explain movement/Internal Merge (Chomsky, 1995, 274ff)[4], binding (Rooryck & Wyngaerd, 2011), External Merge (Wurmbrand, 2014), among many other phenomena. Indeed, it is difficult to find a single phenomenon that falls under the umbrella of syntax which has not been given an Agree-based analysis.

This proliferation of theories of Agree is further exacerbated by the fact that, since its inception, analyses in Generative Grammar have always had both derivational and representational expressions. In the theory used in *Aspects* (Chomsky, 1965), for instance, (1) can be given three formal expressions—one derivational expression in (2), and two representational expressions in (3) and (4).

(1) Sincerity may frighten the boy.

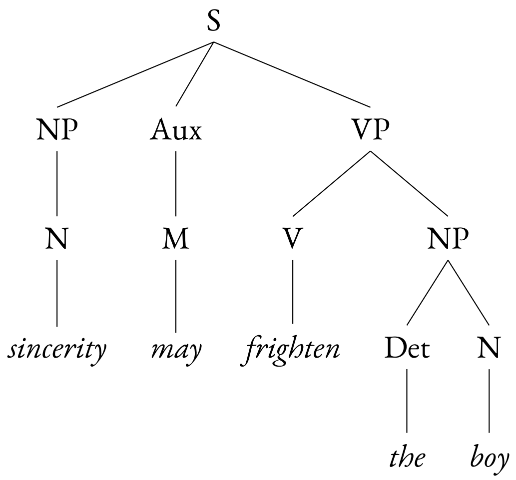
(2)	S	(by S → NP⌢Aux⌢VP)	(cf Chomsky, 1965, p. 68)
	NP⌢Aux⌢VP	(by VP→V⌢NP)	
	NP⌢Aux⌢V⌢NP	(by NP→Det⌢N)	
	NP⌢Aux⌢V⌢Det⌢N	(by NP→N)	
	N⌢Aux⌢V⌢Det⌢N	(by Det→*the*)	
	N⌢Aux⌢V⌢*the*⌢N	(by Aux→M)	
	N⌢M⌢V⌢*the*⌢N	(by M→*may*)	
	N⌢*may*⌢V⌢*the*⌢N	(by N→*sincerity*)	
	sincerity⌢*may*⌢V⌢*the*⌢N	(by N→*boy*)	
	sincerity⌢*may*⌢V⌢*the*⌢*boy*	(by V→*frighten*)	
	sincerity⌢*may*⌢*frighten*⌢*the*⌢*boy*	(by V→*frighten*)	

(3) $[_{\text{S}}\ [_{\text{NP}}\ \textit{Sincerity}_{\text{N}}\ ]\ [_{\text{Aux}}\ \textit{may}_{\text{M}}\ ]\ [_{\text{VP}}\ \textit{frighten}_{\text{V}}\ [_{\text{NP}}\ [_{\text{Det}}\ \textit{the}]\ \textit{boy}_{\text{N}}]]]$

---

[4] Chomsky calls the operation Attract in this work.

(4)

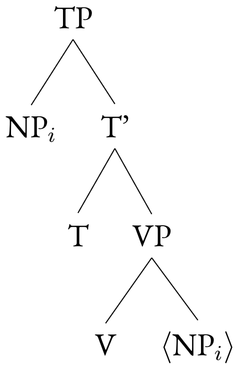


The formal expression in (2) to (4) are all roughly equivalent, though each highlights a different aspect of the analysis they represent.

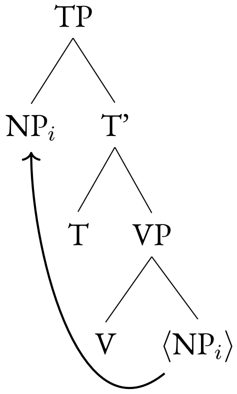
Since Generative Grammar within the P&P tradition is a computational theory, the derivational expression of a given analysis has always been the ultimate expression—a representation is only a valid analysis in such theory, insofar as it can be derived in that theory. The representational expressions, on the other hand, are much more concise and accessible, so they have been overwhelmingly used as shorthands for the derivational expressions, but they are useful as shorthands only insofar as all of the information they encode can also be represented with the derivational expressions.

These representational expressions become problematic, however, when they are augmented for the sake of clarity. For instance movement/Internal Merge can be represented without arrows as in (5), but more often arrows will be added for ease of understanding as in (6), though (5) and (6) are assumed to be equivalent.

(5)

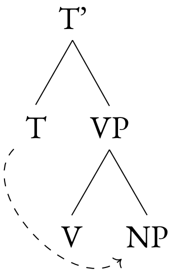


(6)



It is, perhaps, understandable that Agree, commonly represented by arrows similar to movement arrows as in (7), is assumed to have the same level of theoretical underpinning as movement.

(7)



To date, though, there has been no proposal for a derivational expression of the arrow in (7). The task of this paper in part, then, is to remedy this oversight.

To that end, I will be expanding the formalization of minimalist syntax developed by Collins and Stabler (2016). I sketch out this formalization, which is based on a more-or-less contemporary theory within the minimalist program, in section 2, and extend it to include Agree in section 3. While I focus on what I call Long-Distance Downward Valuing (LDDV) Agree, I also discuss how my definitions could be adjusted to reflect other theories such as those that assume feature checking or upward valuation, as well as local varieties of Agree. In section 4 I consider the theoretical implications of my definition of Agree, including its relation to Merge, its implications for the Lexicon, and its relation to the No Tampering Condition. Finally, in section 6 I give some concluding remarks.

2 What does a definition look like?

Collins and Stabler (2016) provide a framework for formal definition. This formal definition uses sets and their basic predicates, relations, and operations (membership, subset, set difference, etc) and finite sequences referred to as "pairs," "triples," and so on depending on their size. Using these formal notions, the grammar they define is such that a number of organizing principles of minimalist theories are provable as theorems of this system. I will be defining Agree in this framework, and in order to understand what it means to define a derivational operation, I must first lay out some basic definitions starting with Universal Grammar (UG) in (8).

(8) Universal Grammar is a 6-tuple: ⟨PHON-F, SYN-F, SEM-F, Select, Merge, Transfer⟩

PHON-F, SYN-F, and SEM-F are universal sets of phonetic, syntactic, and semantic features, respectively; Select, Merge, and Transfer are operations. I will begin the outline of the formal grammar with the feature sets, postponing discussion of the operations for now. Collins and Stabler (2016) (hereafter C&S) also define the set PHON-F* as the set of all possible phonetic strings. These feature-sets are grouped together to form lexical items, which are grouped into a lexicon, which effectively defines individual grammars, as in (9)–(11).[5]

(9) A lexical item is a triple: LI = ⟨PHON, SYN, SEM⟩
where SEM and SYN are finite sets such that SEM $\subset$ SEM-F, SYN $\subset$ SYN-F, and PHON $\in$ PHON-F*.

(10) A lexicon is a finite set of lexical items.

(11) An I-Language is a pair ⟨Lex, UG⟩, where Lex is a lexicon and UG is Universal Grammar.

In order to capture the Copy/Repetition distinction, C&S introduce lexical item tokens, defined in (12), which are the atoms of syntactic computation. C&S, also define several other useful terms using LI tokens.[6]

---

[5] The grammar C&S formalize seems to assume an "early-insertion" theory of morphology. Under a "late-insertion" theory of morphology (Halle & Marantz, 1993; Starke, 2010), LIs would be pairs of syntactic and semantic features ⟨SYN, SEM⟩. While such a move would likely require C&S to reformulate Transfer, it will be largely irrelevant to the task at hand.

[6] See Collins and Groat (2018) for a survey of the various approaches to capturing the Copy/Repetition distinction.

(12) A lexical item token is a pair: $LI_k = \langle LI, k \rangle$, where LI is a lexical item, and k is an integer.

(13) A lexical array is a finite set of lexical item tokens.

(14) X is a syntactic object iff:

i. X is a lexical item token, or

ii. X is a set of syntactic objects.

(15) Let A and B be syntactic objects, then B immediately contains A iff $A \in B$.

(16) Let A and B be syntactic objects, then B contains A iff

i. B immediately contains A, or

ii. for some syntactic object C, B immediately contains C and C contains A.

C&S then define a generative framework, wherein complex syntactic objects are derived in stages.

(17) A stage is a pair $S = \langle LA, W \rangle$, where LA is a lexical array [a possibly ordered set of lexical item tokens] and W is a set of syntactic objects. We call W the workspace of S.

The operations Merge, Select, and Transfer operate on stages and derive new stages. Merge is binary set-formation, Select moves lexical item tokens from the lexical array to the workspace[7], and Transfer converts syntactic objects into interface objects. Merge and Select are rather simple, as shown in (18) and (19). Transfer, on the other hand, is more complicated—so much so that C&S devote 5 sections of their paper to developing its definition. Since Transfer is not strictly relevant to this paper, I will omit its definition.

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

(19) Let S be a stage in a derivation $S = \langle LA, W \rangle$.

If lexical token $A \in LA$, then $\text{Select}(A, S) = \langle LA - \{A\}, W \cup \{A\} \rangle$

Thus, we can define the central notion of derivation in (20)

---

[7] The operation Select is not to be confused with (c-/s-)selection. The first, indicated by capitalization, is a purely formal/theoretical construct, while the latter is an empirical generalization about categorial/semantic restrictions on phrase structure.

(20) A derivation from lexicon L is a finite sequence of stages $\langle \mathrm{S}_1, \ldots, \mathrm{S}_n \rangle$, for $n \geq 1$, where each $\mathrm{S}_i = \langle \mathrm{LA}_i, \mathrm{W}_i \rangle$, such that

   i. For all LI and k such that $\langle \mathrm{LI,k} \rangle \in \mathrm{LA}_1$, $\mathrm{LI} \in \mathrm{L}$,

   ii. $\mathrm{W}_1 = \{\}$ (the empty set),

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

      (derive-by-Select) for some $\mathrm{A} \in \mathrm{LA}$ i , $\langle \mathrm{LA}_{i+1}, \mathrm{W}_{i+1} \rangle = \mathrm{Select}(\mathrm{A}, \langle \mathrm{LA}_i, \mathrm{W}_i \rangle)$, or

      (derive-by-Transfer) ..., or

      (derive-by-Merge) $\mathrm{LA}_i = \mathrm{LA}_{i+1}$, and the following conditions hold for some A,B:

      a. $\mathrm{A} \in \mathrm{W}_i$

      b. Either A contains B [Internal Merge] or W_i immediately contains B [External Merge], and

      c. $\mathrm{W}_{i+1} = (\mathrm{W}_i - \{\mathrm{A,B}\}) \cup \{\mathrm{Merge(A,B)}\}$

So, abstracting away from certain representational complexities, the sentence *Brian smiles* would be derived as in (21).

(21) (S$_1$) $\langle \{\mathrm{T}_{\mathrm{Pres}}, \textit{smile}, \textit{Brian}\}_{\mathrm{LA}_1}, \{\}_{\mathrm{W}_1} \rangle$ (by Select(*Brian*, S$_1$))

(S$_2$) $\langle \{\mathrm{T}_{\mathrm{Pres}}, \textit{smile}\}_{\mathrm{LA}_2}, \{\textit{Brian}\}_{\mathrm{W}_2} \rangle$ (by Select(*smile*, S$_2$))

(S$_3$) $\langle \{\mathrm{T}_{\mathrm{Pres}}\}_{\mathrm{LA}_3}, \{\textit{Brian}, \textit{smile}\}_{\mathrm{W}_3} \rangle$ (by Merge(*smile*, *Brian*))

(S$_4$) $\langle \{\mathrm{T}_{\mathrm{Pres}}\}_{\mathrm{LA}_4}, \{\{\textit{Brian}, \textit{smile}\}\}_{\mathrm{W}_4} \rangle$ (by Select([Pres], S$_4$))

(S$_5$) $\langle \{\}_{\mathrm{LA}_5}, \{\mathrm{T}_{\mathrm{Pres}}, \{\textit{Brian}, \textit{smile}\}\}_{\mathrm{W}_5} \rangle$ (by Merge([Pres], {*smile*, *Brian*}))

(S$_6$) $\langle \{\}_{\mathrm{LA}_6}, \{\{\mathrm{T}_{\mathrm{Pres}}, \{\textit{Brian}, \textit{smile}\}\}\}_{\mathrm{W}_6} \rangle$ (by Merge(*Brian*, $\{\mathrm{T}_{\mathrm{Pres}}, \ldots\}$))

(S$_7$) $\langle \{\}_{\mathrm{LA}_7}, \{\{\textit{Brian}, \{\mathrm{T}_{\mathrm{Pres}}, \{\textit{Brian}, \textit{smile}\}\}\}\}_{\mathrm{W}_7} \rangle$

C&S's formalization is open for some refinements, such as those that Chomsky (2020) suggests, and extensions, but it provides us with a framework for those refinements and extensions. In order to add Agree to the formal grammar, for instance, we would need to define it as a function from stages to stages to be added as a derive-by-Agree clause to (20), and in order to define such a function, as we shall see, we will need a formal definition of features.

3 Defining Agree

Agree can be very broadly described as an operation that modifies a syntactic object X iff X stands in a particular formal/structural relation and a particular substantive relation with another syntactic object Y. So, in order to define Agree, we must formalize (*a*) the formal/structural prerequisite—Probe, a species of Search—(*b*) the substantive prerequisite—Match—and (*c*) the process of modifying the syntactic object in question—Value or Check—each of which has, in a sense, been the focus of its own debate in the literature. As a starting point, I will formalize Long-Distance Downward Valuation Agree (LDDV-Agree), which is more or less the version of Agree put forth by Wurmbrand (2014) and which has the following properties. LDDV-Agree is long-distance in that it does not require a strictly local relation between the Agreeing syntactic objects, rather two elements stand in a c-command-plus-relativized-minimality relation as specified in (22).[8]

(22) Two elements X and Y can Agree iff X c-commands Y, Y Matches X, and there is no element H such that H Matches X, X c-commands H and H c-commands Y.

LDDV-Agree is downward in the sense that it modifies the c-commanded element, and it is valuation-based in the sense that the element is modified by converting one of its unvalued feature into a valued one as specified in (23) and (24).

(23) X Matches Y for feature F iff X has [F:*val*] and Y has [F:__].[9]

(24) If X and Y Agree for feature F then [F:__] on Y becomes [F:*val*].

The first thing we must do, is formalize the notion of "feature" as used here. By (8), there are three sets of features in Universal Grammar—PHON-F, SYN-F, SEM-F. Setting aside PHON-F as irrelevant to the current paper, our task is to formalize the members of SYN-F and SEM-F. Generally, a given syntactic or semantic feature is describable with reference to its interpretability, its type, and its value (or lack thereof).

---

[8] The two elements participating in Agree are commonly referred to as the probe and the goal respectively. The term "probe" is also often used to refer to the search process associated with Agree. To avoid this confusing ambiguity, I do not use "probe" and "goal" to refer to elements.

[9] Multiple commentators have noted that a more intuitive and simple definition of Match would allow an X with [F:*val*] to Match a Y with [F:*val*]. Such a definition, though would be inconsistent with the contemporary theories of Agree that are being formalized here—theories in which Agree is the process by which an element with a *valued* feature values an *unvalued* feature on different element.

Interpretability can be taken care of by simple set membership—interpretable features are members of SEM-F, uninterpretable features are members of SYN-F—leaving us with type and value.[10] Keeping with Wurmbrand (2014) as our basis, then, we can define features as in (25) along with a few auxiliary notions defined in (26) to (28).[11]

(25) A *feature* is a pair $\langle F, v\rangle$—hereafter abbreviated F_v—where v is an integer. F is called the *feature type*, v is the *feature value*.

(26) For all feature types F, $\langle F, 0\rangle$ is an *unvalued* F feature.

(27) For lexical item LI = $\langle$PHON, SYN, SEM$\rangle$, feature F_v is a *feature of* LI, iff $F_v \in$SYN or $F_v \in$SEM.

(28) For lexical item token $LI_k = \langle LI, k\rangle$, feature F_v is a *feature of* LI_k, iff F_v is a feature of LI.

So, for instance, English present tense might have roughly the lexical representation in (29).

(29) $\langle \text{PHON}, \{\ldots\langle\varphi, 0\rangle\ldots\}, \{\ldots\langle \text{T}, 1\rangle\ldots\}\rangle$

This lexical item has some phonetic features, an unvalued uninterpretable φ-feature, and an interpretable T feature with the value 1, which we can stipulate is interpreted as present tense. The choice to formalize feature values as integers is made only to allow for a perspicuous way of defining unvalued features. We could use any type of discrete symbol to represent values, provided it had a special symbol for "unvalued."

We can define Match as in (30).

(30) For any two lexical item tokens X, G and feature type F,
Match(X, G, F) = 1 iff for some feature value $v \neq 0$, $\langle F, v\rangle$ is a feature of X and $\langle F, 0\rangle$ is a feature of G.

Under this definition, an English finite T head will match a non-Case-marked pronoun but not a Case-marked one, as demonstrated in (31)

---

[10] The fact that SYN-F and SEM-F seem to be disjoint subsets of a natural class of features seems to indicate that they are not independent of each other. Indeed, the Strong Minimalist Thesis (SMT) would say that there is only one set of features in UG—SEM-F. This, of course raises a number of fascinating questions which are beyond the scope of this paper.

[11] An anonymous reviewer points out that, although Wurmbrand (2014) represents features as name-value pairs, they are more commonly assumed to be organized into hierarchical feature geometries (Béjar, 2003; Harbour, 2007; Harley & Ritter, 2002). In section 5 discuss the formalization of one such feature theory and its limited effect on the overall formal definition of Agree.

(31) a. Match(T, $3\mathrm{SgF}_{[\mathrm{Case:\_}]}$, Case) =

$$\mathrm{Match}\left(\begin{array}{l}\langle\langle \mathrm{PHON_T}, \mathrm{SYN_T}, \{\ldots, \langle \mathrm{Case}, 1\rangle, \ldots\}\rangle, k\rangle, \\ \langle\langle \mathrm{PHON_{3sgF}}, \{\ldots, \langle \mathrm{Case}, 0\rangle, \ldots\}, \mathrm{SEM_{3SgF}}\rangle, k\rangle, \mathrm{Case}\end{array}\right) = 1$$

b. Match(T, $3\mathrm{SgF}_{[\mathrm{Case:ACC}]}$, Case) =

$$\mathrm{Match}\left(\begin{array}{l}\langle\langle \mathrm{PHON_T}, \mathrm{SYN_T}, \{\ldots, \langle \mathrm{Case}, 1\rangle, \ldots\}\rangle, k\rangle, \\ \langle\langle \mathrm{PHON_{3sgF}}, \{\ldots, \langle \mathrm{Case}, 2\rangle, \ldots\}, \mathrm{SEM_{3SgF}}\rangle, k\rangle, \mathrm{Case}\end{array}\right) = 0$$

Value is essentially a replacement operation—operating on a lexical item token, swapping an unvalued feature with a valued counterpart. This is defined in (32).

(32) For lexical item token $\mathrm{LI}_k = \langle\langle \mathrm{PHON}, \mathrm{SYN}, \mathrm{SEM}\rangle, \mathrm{k}\rangle$, and feature $\langle \mathrm{F}, \mathrm{v}\rangle$,

$\mathrm{Value}(\mathrm{LI}_k, \langle \mathrm{F}, \mathrm{v}\rangle) = \langle\langle \mathrm{PHON}, (\mathrm{SYN} - \{\langle \mathrm{F}, 0\rangle\}) \cup \{\langle \mathrm{F}, \mathrm{v}\rangle\}, \mathrm{SEM}\rangle, \mathrm{k}\rangle$

So, an instance of Value associated with subject-predicate agreement, ignoring Case, might look something like (33).

(33) Where $\mathrm{T_{Pres}}$ = (29) and $\langle \varphi, 31\rangle$ corresponds to 3rd person singular,

$\mathrm{Value}(\langle \mathrm{T_{Pres}}, 4\rangle, \langle \varphi, 31\rangle) \rightarrow \langle\langle \mathrm{PHON}, \{\ldots \langle \varphi, 31\rangle \ldots\}, \{\ldots \langle \mathrm{T}, 1\rangle \ldots\}\rangle, 4\rangle$

The resulting lexical item token still has an interpretable tense feature and an uninterpretable φ feature but the latter now has the value 31, which we stipulate corresponds to 3rd person singular.

Note that, while I have been tacitly assuming that (un)valued-ness and (un)interpretability are correlated in the lexicon—that all and only unvalued features are members of SYN-F—the definition of Value in (32) contradicts this assumption, since the result of Value is an element that contains at least one valued uninterpretable feature.

In fact, any attempt to make the assumption that all uninterpretable features are unvalued hold in general runs into issues. We could save it by eliminating Value, but this would contradict one of the core premises of Agree theory—that Agree modifies lexical item tokens mid-derivation. Alternatively, We could save it by re-defining Value, say as Value′ in (34) which removes the unvalued feature from SYN and adds a valued feature to SEM.

(34) For lexical item token $\mathrm{LI}_k = \langle\langle \mathrm{PHON}, \mathrm{SYN}, \mathrm{SEM}\rangle, \mathrm{k}\rangle$, and feature $\langle \mathrm{F}, \mathrm{v}\rangle$,

$\mathrm{Value}'(\mathrm{LI}_k, \langle \mathrm{F}, \mathrm{v}\rangle) = \langle\langle \mathrm{PHON}, \mathrm{SYN} - \{\langle \mathrm{F}, 0\rangle\}\rangle, \mathrm{SEM} \cup \{\langle \mathrm{F}, \mathrm{v}\rangle\}\rangle, \mathrm{k}\rangle$

With this definition though, our subject-predicate agreement would look like (35), which seems to create a T head which is *semantically* 3rd person singular—something that does not exist, at least in English.

(35) Where $\mathrm{T}_{\mathrm{Pres}}$ = (29) and $\langle \varphi, 31 \rangle$ corresponds to 3rd person singular,
$\mathrm{Value}'(\langle \mathrm{T}_{\mathrm{Pres}}, 4\rangle, \langle \varphi, 31\rangle) \rightarrow \langle\langle \mathrm{PHON}, \{\dots\}, \{\dots \langle \mathrm{T}, 1\rangle, \langle \varphi, 31\rangle \dots\}\rangle, 4\rangle$

The operation defined in (32), then, seems to match the notion of valuation generally assumed in theories of UG with Agree. It does, however have a problematic prediction that I address in section 4.3.

The last portion of Agree to be defined is what is often called "Probe", which is an instance of "Minimal Search" (Chomsky, 2004) an algorithm that requires some discussion.

3.1 Minimal Search

The term Minimal Search, as its usually used in minimalist syntactic theory, refers to an algorithm that retrieves the "highest" object in a structure that meets some particular criterion. While such an algorithm is almost certainly required for Agree, it is not required only for Agree. Indeed, Minimal Search is implicated in at least Internal Merge (Chomsky, 2020) and labelling (Chomsky, 2013).

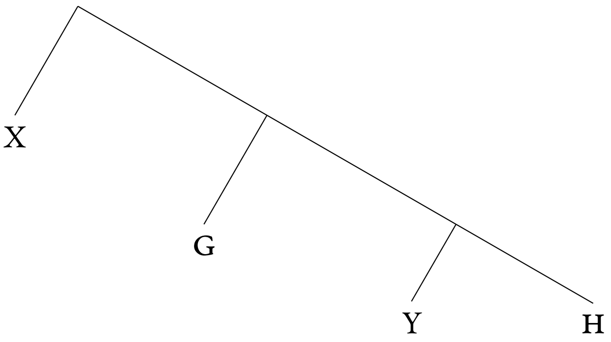
The criterion for a given instance of Search, it seems, depends on the purpose of that search. For Internal Merge, following a Free-Merge theory, the Search criterion is more or less identity—Internal Merge of X and Y requires a successful Search of X for Y or vice versa—Chomsky's (2013) Labelling Algorithm Searches for any lexical item token, and a Search in service of Agree will use Match as defined in (30) as its criterion. Thus, our definition of Search, while guided by the present goal of formalizing Agree, must be general.

In order to properly define a Minimal Search algorithm we must first consider some test cases as follows. Each case is a complex abstract syntactic object containing two objects—G and H—each of which meets the search criterion. Each case is represented both as a binary set as constructed by Merge and a binary tree. The first case in (36) is the most straightforward—G asymmetrically c-commands H, so Minimal Search retrieves G and not H.

(36) **Case 1:** G is retrieved.

a. $\{X, \{G, \{Y, H\}\}\}$

b.

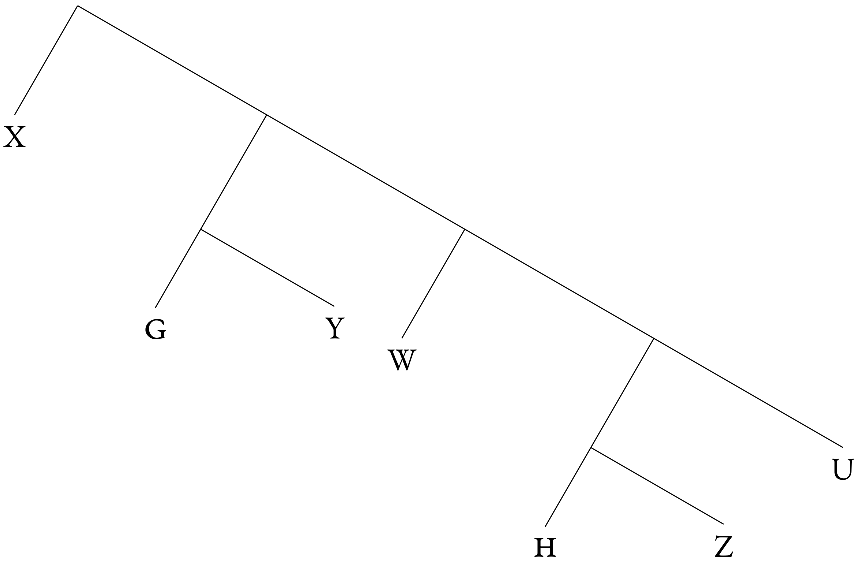


The second case in (37) is slightly more complicated—G does not c-command H, but Minimal Search should retrieve G because it is immediately contained in an object that asymmetrically c-commands an object that immediately contains H.

(37) **Case 2:** G is retrieved.

a. $\{X, \{\{G, Y\}, \{W, \{\{H, Z\}, U\}\}\}\}$

b.

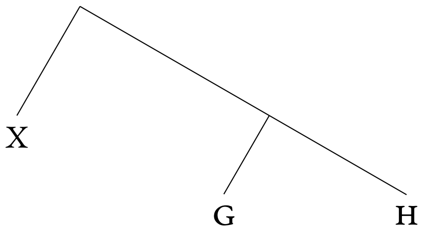


Other cases, though, will give ambiguous results. These are cases in which G and H are equidistant from the root. In (38), for instance G and H are siblings, while in (39) they are immediately contained, respectively, by siblings.

(38) **Case 3:** Both G and H are retrieved.

a. $\{X, \{G, H\}\}$

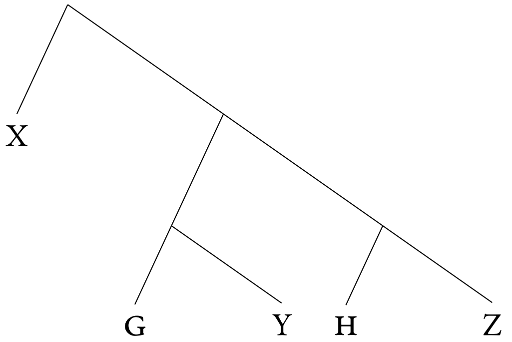
b.



(39) **Case 4:** Both G and H are retrieved.

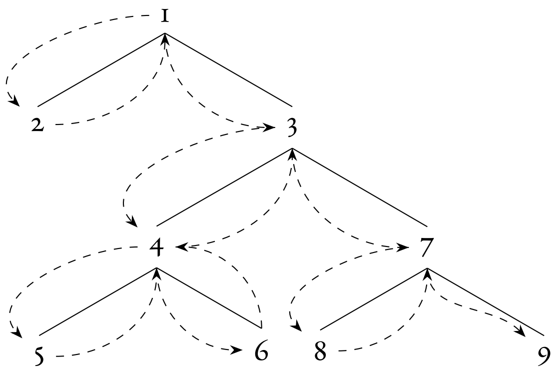
a. $\{X\{\{G, Y\}, \{H, Z\}\}\}$

b.



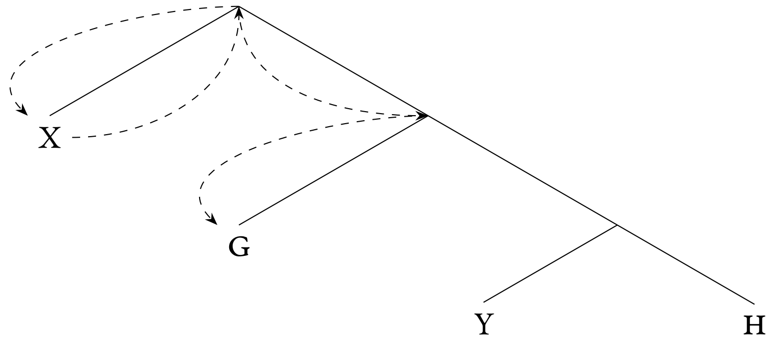
Our goal, then, is to construct an algorithm that has the above-defined results. There are two broad classes of search algorithms appropriate to our task—Depth-First Search (DFS) and Breadth-First Search (BFS). DFS starts at the root of an object and searches to a terminal node before backtracking, as represented in (40), where the arrows and the numbers indicated the search order.

(40)



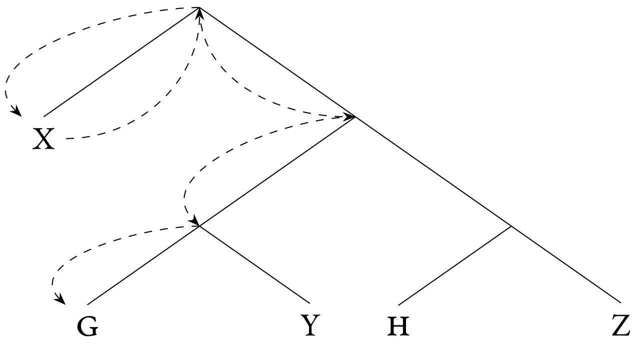
A DFS algorithm can be made minimal by designing it to stop as soon as it finds a node that meets its criterion. So, a Minimal DFS on Case 1 would be proceed as in (41) selecting.

(41)



However in an ambiguous case, like Case 4, a Minimal DFS will incorrectly retrieve just a single object as shown in (42).

(42)



A Minimal DFS algorithm, then is over-definite—it gives a definite result where we expect an ambiguous one.

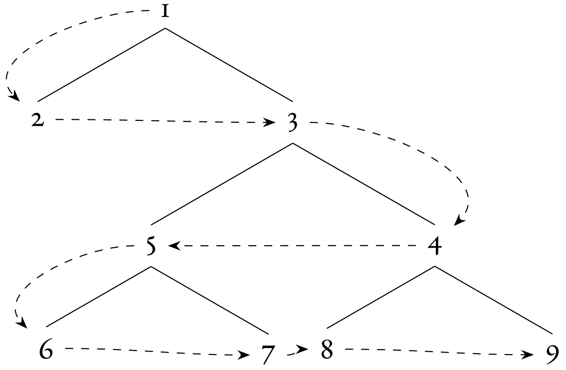
There is also a deeper problem with DFS as applied to syntactic objects, and that is its reliance on linear order as well as structure. In the examples above, whenever the algorithm reaches a branching node, it takes the left branch first. If it, instead, took the right branch first, the result would be different—in both (41) and (42), a right-to-left Minimal DFS would retrieve H rather than G. The problem is made worse by the fact that, the structures that we are searching are constructed by Merge and, therefore, do not have a linear order. In order for our algorithm to make a decision at a "branch," then, it would have to be a random decision. Therefore, the result of a DFS for a given syntactic object may be different each time it is run. Given these issues, I will set aside DFS.[12]

---

[12] While Branan and Erlewine (forthcoming), Ke (2019), and Preminger (2019) all make reference to the issues with a minimal DFS, none opt for a BFS, with Preminger and Ke each defining a version of DFS and Branan and Erlewine making no firm decision between the two options. Branan and Erlewine and Preminger both argue that the weaknesses of DFS can be avoided if certain parts of a structure are inaccessible to Search, however neither provide a principled way of so restricting the DFS algorithm. Preminger proposes that specifiers are not searched, while Branan and Erlewine suggest that left-branches might

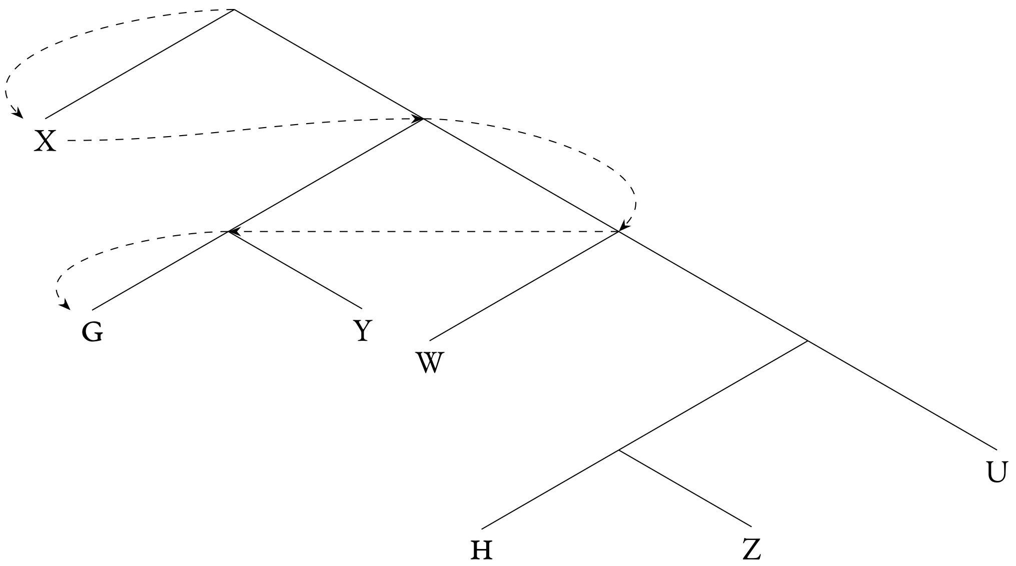
Breadth-first Search (BFS) algorithms, on the other hand, searches neighbour nodes before proceeding lower in the tree as represented in (43), where the arrows and the numbers indicated the search order.

(43)



Again, this can be made minimal by requiring that the algorithm stop immediately upon finding an object that matches the search criterion. A Minimal BFS on Case 2, then, is represented in (44).

(44)



Like the Minimal DFS, the Minimal BFS, as represented in (43) and (44) assumes that nodes are linearly ordered, even if that order is arbitrary. Unlike the Minimal DFS, the order of the neighbour nodes does not matter, at least for definite cases like Case 1 and Case 2. To demonstrate this, consider the reverse

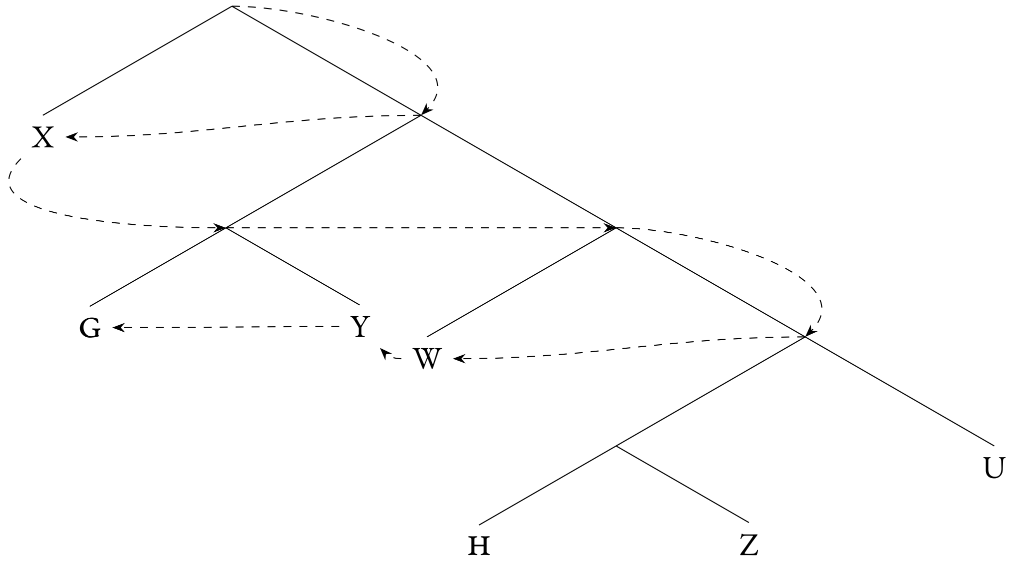
---

not be searched. Both of these proposals, though, depend on an assumption that syntactic objects produced by Merge are inherently asymmetric, while the present paper assumes the exact opposite.

Ke (2019, pp. 46–49), on the other hand, claims to propose a BFS algorithm but, in fact, proposes a parallelized DFS. This solves the issue of the unordered nature of syntactic objects—when faced with two "branches" the algorithm does not need to make a choice, it searches both simultaneously. Unfortunately, Ke is not explicit about his model of parallel computation. Specifically, he does not define how multiple processes running in parallel are able to communicate with each other so that, say, one process can report success and cause the overall process to halt.

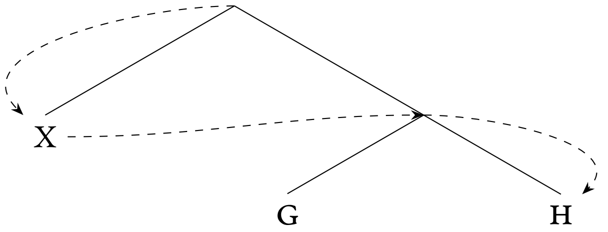
version of (44) in (45).

(45)

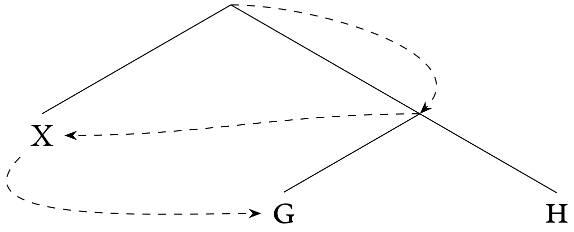


In an ambiguous case, though, Minimal BFS suffers the same fate as Minimal DFS—it is over-definite. So, in Case 3, Minimal BFS will wrongly retrieve either G or H depending on the ordering of nodes, as shown in (46) and (47).

(46)

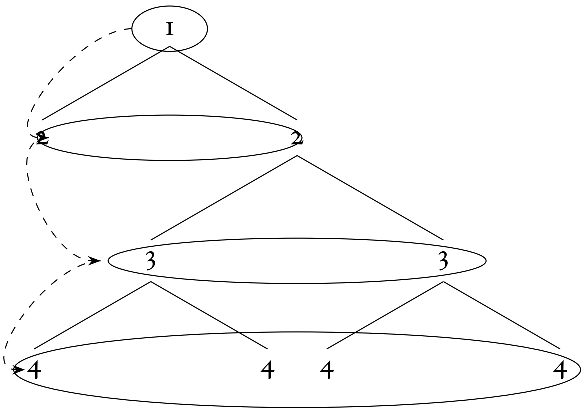


(47)



This flaw, however, can be overcome if, instead of traversing each node, we treat the sets of neighbour nodes as tiers, as in (48).

(48)



Minimal Tiered BFS, then, would visit each tier and extract the subset of that tier whose members all matched the search criterion, and stop as soon at it extracts a non-null subset. Thus we can define a definite search result as in (49), an ambiguous search result as in (50), and a failed search as in (51).

(49) For a syntactic object SO and criterion P, Search(SO,P) is definite iff $|\text{Search(SO,P)}|=1$

(50) For a syntactic object SO and criterion P, Search(SO,P) is ambiguous iff $|\text{Search(SO,P)}| > 1$

(51) For a syntactic object SO and criterion P, Search(SO,P) is failed iff $\text{Search(SO,P)} = \{\}$

Minimal Tiered BFS, then, will be our choice of Search algorithm. The next step is to formally define it.

In order to define Search, then, we need to be able to properly generate search tiers. So, for instance, the tiers for (37) are given in (52)

(52) $\text{Tier 1} = \{X, \{\{G, Y\}, \{W, \{\{H, Z\}, U\}\}\}\}$

$$\text{Tier 2} = \left\{ \begin{array}{l} \{G, Y\}, \\ \{W, \{\{H, Z\}, U\}\} \end{array} \right\}$$

$$\text{Tier 3} = \left\{ \begin{array}{l} G, Y, \\ \{\{H, Z\}, U\} \\ W \end{array} \right\}$$

$$\text{Tier 4} = \left\{ \begin{array}{l} \{H, Z\}, \\ U \end{array} \right\}$$

$\text{Tier 5} = \{H, Z\}$

$\text{Tier 6} = \{\}$

For a given Tier T_i, we can generate T_{i+1} by first removing all the terminal nodes from T_i and performing what is called an arbitrary union which is defined in (53).

(53) For a set of sets $\bar{X} = \{X_0, \ldots, X_n\}$ the arbitrary union of $\bar{X}$, $\bigcup \bar{X} = X_0 \cup \cdots \cup X_n$.

Therefore we can define a procedure NextTier in (54) and with it, Search in (55).

(54) For T, a set of syntactic objects,

NextTier(T)= $\bigcup\{\text{SO}\in\text{T: SO is not a lexical item token}\}$.

(55) For S, a set of syntactic objects, and Crit, a predicate of lexical item tokens,

$$\text{Search(S,Crit)} = \begin{cases} \{\} \text{ if S} = \{\} \\ \{\text{SO} \in \text{S} : \text{Crit(SO)} = 1\} \text{ if } \{\text{SO} \in \text{S} : \text{Crit(SO)} = 1\} \neq \{\} \\ \text{Search(NextTier(S), Crit) otherwise} \end{cases}$$

Probe, then is a special type of Search, where the search criterion is based on Match as shown in (56).

(56) For F, a feature type, and SO, a syntactic object that immediately contains X, a lexical item token,

Probe(SO,X,F) = Search(SO, $\text{Match}^{\text{X,F}}$)

where $\text{Match}^{\text{X,F}}$ = 1 iff X contains a feature g such that Match(X, g, F) = 1.[13]

With our definition of Probe in place, we can turn to our final definition of Agree which I turn to shortly in section 3.2.

3.2 A formal definition of Agree

If and when an instance of Probe retrieves a lexical item token, that token must be modified—at least according to most versions of Agree.[14] More precisely, the token must be modified in place. That is, if token G is in position Q in stage S_i, then the modified token G′ must be in position Q in stage S_{i+1}. Furthermore, if copies of G are in multiple positions (Q, Q′, Q″...) in S_i, then copies of X′ must be in

---

[13] $\text{Match}^{\text{X,F}}$ can be formally defined using the technique of λ-abstraction as $(\lambda g.(\text{Match}(\text{X}, g, \text{F})))$. See Church (1941) or Partee et al. (1990) for an introduction to the lambda calculus.

[14] If we wished to define Agree purely as a relation—*i.e.* an n-place predicate (n>1)—we could simply define it as Agree_{pred}(SO, X, Y, F) iff Probe(SO, X, F) = Y.

those same positions in S_{i+1}. In order to do this we must traverse the syntactic object in question and replace every instance of G with G′, the result of Value.

Note that each copy of G must be replaced to maintain their copy-hood. Taking, for example, the pronoun *her* in (57), which has at least two copies as indicated in (58).

(57) We expect her to be hired.

(58) $\left\{\ldots\left\{\text{Voice},\left\{expect,\left\{\text{3SgF}_{[\text{Case:\_}]},\left\{\ldots\left\{hire,\text{3SgF}_{[\text{Case:\_}]}\right\}\ldots\right\}\right\}\right\}\right\}\ldots\right\}$

If accusative Case marking was performed by a "minimal" Agree—one that only valued the highest copy—then the result would be the syntactic object in (59) in which the two instances of the third person feminine pronoun are distinct from each other and, therefore, no longer copies in any sense.

(59) $\left\{\ldots\left\{\text{Voice},\left\{expect,\left\{\text{3SgF}_{[\text{Case:ACC}]},\left\{\ldots\left\{hire,\text{3SgF}_{[\text{Case:\_}]}\right\}\ldots\right\}\right\}\right\}\right\}\ldots\right\}$

The pronouns $\text{3SgF}_{[\text{Case:ACC}]}$ and $\text{3SgF}_{[\text{Case:\_}]}$ are clearly distinct—one is Case-marked, the other isn't—and furthermore, they have divergent derivational histories—one has undergone Value, the other hasn't. What's more is that the lower pronoun is not Case-marked and should therefore cause a crash at the interfaces. In order to maintain the identity between copies, then, Agree must be maximal—it must Value every copy.

Thus we can define Agree as in (60).

(60) Where SO is a syntactic object F is a feature type, and v is a feature value $\neq 0$ and G is a lexical item token such that Probe(α, X, F_v) = {G}, where SO = αor SO is contained in α

$$\text{Agree}(\text{SO}, \text{G}, \text{F}_\text{v}) = \begin{cases} \text{Value}(\text{SO},\langle\text{F, v}\rangle) \text{ if SO=G} & (a) \\ \text{SO if SO is a lexical item token} & (b) \\ \{\text{Agree}(\text{A, G, F}_\text{v}), \text{Agree}(\text{B, G, F}_\text{v})\} \text{ if SO} = \{\text{A,B}\} & (c) \end{cases}$$

Agree, according to (60), is defined for three cases. In Case (60a), where SO is an instance of G—the lexical item token to be valued, the output of Agree is the valued version of G—Agree applies non-vacuously. In Case (60b), where SO is a lexical item token, but not an instance of G, the output of Agree is SO—Agree applies vacuously. In Case (60c), where SO is a set, Agree is applied to each member of SO, and a new set containing the respective outputs of those Agree operations is constructed—Agree applies recursively. Note also, that the result of Case (60c)—binary set-formation—is an instance of Merge, and I will treat it as such below.

To see how Agree works, consider accusative Case marking in the sentence *Brian kisses him* as an instance of Agree operating on the structure in (61) yielding the structure in (62).

(61) $\left\{_{\alpha}\text{Voice}, \left\{_{\beta}\textit{kiss}, \text{3SgM}_{[\text{Case:\_}]}\right\}\right\}$

a. $\text{Voice} = \langle\langle \text{PHON}_{\text{Voice}}, \text{SYN}_{\text{Voice}}, \{\ldots, \langle \text{Case}, 2\rangle, \ldots\}\rangle, k\rangle$

(Voice contains an Accusative Case feature in its SEM)

b. $\text{3SgM}_{[\text{Case:\_}]} = \langle\langle \text{PHON}_{\text{3SgM}}, \{\ldots, \langle \text{Case}, 0\rangle, \ldots\}, \text{SEM}_{\text{3SgM}}\rangle, k'\rangle$

(The 3rd person singular masculine pronoun contains an unvalued Case feature in its SYN)

(62) $\left\{_{\alpha}\text{Voice}, \left\{_{\beta}\textit{kiss}, \text{3SgM}_{[\text{Case:ACC}]}\right\}\right\}$

The first step of this instance of Agree is to Probe for unvalued Case features, as in (63)

(63) $\text{Probe}(\alpha, \text{Voice}, \text{Case}) = \left\{\text{3SgM}_{[\text{Case:\_}]}\right\}$

The non-case-marked pronoun—*i.e.*, the sole member of the result of Probe—stands in for G in (60) for our instance of Agree. Since α is a complex SO, the first instance of Agree, as shown in (64), proceeds by recursively performing Agree on α's constituent parts—Voice and β—and Merging the results. Since Voice is a lexical item token but not our target for Agree, Agree does not change it, as shown in (65), and we can simplify our first iteration of Agree as in (66).

(64) $\text{Agree}(\alpha, \text{3SgM}_{[\text{Case:\_}]}, ACC) =$ (by (60c))

$\text{Merge}(\text{Agree}(\text{Voice}, \text{3SgM}_{[\text{Case:\_}]}, ACC), \text{Agree}(\beta, \text{3SgM}_{[\text{Case:\_}]}, ACC))$

(65) $\text{Agree}(\text{Voice } \text{3SgM}_{[\text{Case:\_}]}, ACC) = \text{Voice}$ (by (60b))

(66) $\text{Merge}(\text{Agree}(\text{Voice}, \text{3SgM}_{[\text{Case:\_}]}, ACC), \text{Agree}(\beta, \text{3SgM}_{[\text{Case:\_}]}, ACC)) =$ (by (65))

$\text{Merge}(\text{Voice}, \text{Agree}(\beta, \text{3SgM}_{[\text{Case:\_}]}, ACC))$

We then perform Agree on β which contains the verb and the direct object pronoun.

(67) Agree(β, 3SgM$_{[\text{Case:\_}]}$, Case$_{\text{ACC}}$) = (by (60c))
Merge(Agree(*kiss*, 3SgM$_{[\text{Case:\_}]}$, Case$_{\text{ACC}}$), Agree(3SgM$_{[\text{Case:\_}]}$, 3SgM$_{[\text{Case:\_}]}$, Case$_{\text{ACC}}$))

(68) Agree(*kiss*, 3SgM$_{[\text{Case:\_}]}$, Case$_{\text{ACC}}$) = *kiss* (by (60b))

(69) Agree(3SgM$_{[\text{Case:\_}]}$, 3SgM$_{[\text{Case:\_}]}$, Case$_{\text{ACC}}$) = (by (60a))
Value(3SgM$_{[\text{Case:\_}]}$, Case$_{\text{ACC}}$) = (by (32))
3SgM$_{[\text{Case:ACC}]}$

(70) Merge(Agree(*kiss*, 3SgM$_{[\text{Case:\_}]}$, Case$_{\text{ACC}}$), Agree(3SgM$_{[\text{Case:\_}]}$, 3SgM$_{[\text{Case:\_}]}$, Case$_{\text{ACC}}$)) = (by (68), (69))
Merge(*kiss*, 3SgM$_{[\text{Case:ACC}]}$)

Then, having reached the "bottom" of our structure, we are left with two simple Merge operations which yield (62) as shown in (71).

(71) Merge(Voice, Agree(β, 3SgM$_{[\text{Case:\_}]}$, ACC)) = (by (70))
Merge(Voice, Merge(*kiss*, 3SgM$_{[\text{Case:ACC}]}$)) = (by (18))
Merge(Voice, $\left\{ \textit{kiss}, \text{3SgM}_{[\text{Case:ACC}]} \right\}$) = (by (18))
$\left\{ \text{Voice}, \left\{ \textit{kiss}, \text{3SgM}_{[\text{Case:ACC}]} \right\} \right\}$ = (62)

We have arrived at a formal definition of one variety of Agree (LDDV-Agree) which we will use in the the following section as a basis for defining other varieties.

3.3 Upward Valuation

In defining a Downward Valuation Agree, we considered syntactic objects such as the one schematized in (72) which immediately contain lexical item tokens bearing a valued feature F_{v} and which contain a lexical item token bearing an unvalued feature F_0.

(72) $\{\text{X}_{\text{F:v}}, \{\ldots \text{G}_{\text{F:0}}\}\}$

In an Upward Valuation, the relevant features of X and G are swapped, as in (73).

(73) $\{X_{F:0}, \{\ldots G_{F:v}\}\}$

In order to capture Upward Valuation, then we need first modify the Match criterion of Probe as in (74), moving X to the second argument position.

(74) For F, a feature type, and SO, a syntactic object that immediately contains X, a lexical item token, $\text{Probe}_{\text{UV}}(\text{SO}, \text{X}, \text{F}) = \text{Search}(\text{SO}, \text{Match}^{\text{X, F}})$.

Thus, Probe_{UV} gives a definite result {G} only if X contains an unvalued F feature and G contains a valued F feature. Since, by definition, the relevant unvalued feature in Agree_{UV} is at the top of the structure, we might think that no exhaustive DFS is required. Unfortunately, though, the same concern with valuing copies is with us—just because a lexical item token is at the top of a tree doesn't mean there isn't a copy of it at the bottom. Therefore, our definition of Agree_{UV} in (75) look similar to that in (60).

(75) For lexical item token X, syntactic object SO={X, ...}, and feature type F, and lexical item token G such that $\text{Probe}_{\text{UV}}(\alpha, \text{X}, \text{F}_\text{v}) = \{\text{G}\}$, where SO = αor SO is contained in α,

$$\text{Agree}_{\text{UV}}(\text{SO}, \text{X}, \text{F}_\text{v}) = \begin{cases} \text{Value}(\text{SO},\langle \text{F, v}\rangle)\text{if SO=X} & (a) \\ \text{SO if SO is a lexical item token} & (b) \\ \text{Merge}(\text{Agree}_{\text{UV}}(\text{A}, \text{X}, \text{F}_\text{v}), \text{Agree}_{\text{UV}}(\text{B}, \text{X}, \text{F}_\text{v}))\text{ if SO} = \{\text{A,B}\} & (c) \end{cases}$$

3.4 Feature Checking

Versions of Agree whose effects are feature checking rather than valuation assume that all formal features—*i.e.*, members of SYN-F—are valued, but must be checked by Agree (Chomsky, 1995). In order to formalize such a feature checking operation, $\text{Agree}_{\checkmark}$, we must reformulate our notion of features and our Match predicate, and replace Value with Check. Formal features and their related notions, then, are defined as in (76) and (77), with semantic features retaining their definition in (25).

(76) A *formal feature* is a triple ⟨c?, F, v⟩, where c? is 1 or 0 and v is an integer. F is called the *feature type*, v is the *feature value*.

(77) For all feature types F and values v, ⟨ 0, F, v⟩ is an *unchecked* F_v feature, and ⟨ 1, F, v⟩ is *checked* F_v feature.

$\text{Match}_{\checkmark}$, then, compares a semantic feature of one lexical item token with a formal feature of another succeeding if both features have the same type and value and the formal feature is unchecked, as defined in (78)

(78) For any two lexical item tokens X and G, feature type F and value v,

$\text{Match}_{\checkmark}(\text{X, G, F}) = 1$ iff $\langle \text{F, v} \rangle$ is a feature of X and $\langle 0, \text{F, v} \rangle$ is a feature of G.

Finally, Check is a simple matter of flipping a 0 to a 1 or leaving a 1 as a 1 as in (79). Note, though, that Check will never apply to an already checked feature, since Match is a prerequisite for Check and will only succeed if the feature in question is unchecked.

(79) For a lexical item token $\text{SO}=\langle\langle \text{PHON, SYN, SEM} \rangle, k\rangle$, and formal feature $\text{F}_{\text{v}}=\langle \text{c?, F, v} \rangle$,

$\text{Check}(\text{SO}, \text{F}_{\text{v}}) = \langle\langle \text{PHON}, (\text{SYN} - F_{\text{v}}) \cup \{\langle 1, \text{F, v} \rangle\}, \text{SEM} \rangle, k\rangle$

These newly defined functions can be slotted into our formalized definitions of Agree as in (80) to give a definition of $\text{Agree}_{\checkmark}$, where G is the result of Probing based on $\text{Match}_{\checkmark}$.

(80) Where SO is a syntactic object F_{v} is feature, and G is a lexical item token such that $\text{Probe}_{\checkmark}(\alpha, \text{X}, \text{F}_{\text{v}}) = \{\text{G}\}$, where SO = αor SO is contained in α,

$$\text{Agree}(\text{SO, G, F}_{\text{v}}) = \begin{cases} \text{Check}(\text{SO,F}_{\text{v}}) \text{ if SO=G} & (a) \\ \text{SO if SO is a lexical item token} & (b) \\ \text{Merge}(\text{Agree}_{\checkmark}(\text{A, G, F}_{\text{v}}), \text{Agree}_{\checkmark}(\text{B, G, F}_{\text{v}})) \text{ if SO} = \{\text{A,B}\} & (c) \end{cases}$$

3.5 Local Agree

Early minimalist theories of agreement (*e.g.* Chomsky, 1993) continued the GB assumption that agreement was limited to what was called a "spec-head" relation. So, for example, subject-predicate agreement was assumed to occur because, in the terminology of the day, the subject moves to the specifier of the predicate head (T or I), in contrast to later theories in which subjects move because they agree. Similarly, Case licensing, in these theories, is usually taken to occur under a "spec-head" relation. In this section, I will formalize this conception of Agree.

On its surface, Local Agree, as described above, has the advantage of not requiring an arbitrary search of the entire derived expression. Instead, the search is strictly and specifically limited to the very top of object. The canonical case of so-called “spec-head” agreement is the finite subject merged with the finite predicate, shown in (81)

(81) TP = {{D, …}, {T, …}}

Restricting our discussion to Case, we can see that the Agree operation is an interaction between the lexical item token immediately contained in one member of TP and the lexical item token contained in the other member of TP. We can define $\text{Probe}_{\text{Local}}$, then, as in (82).

(82) For feature type F, lexical item tokens X and Y, and syntactic object SO={U, W},

$$\text{Probe}_{\text{Local}}(\text{SO, X, F}) = \begin{cases} \text{Y if X} \in \text{U, Y} \in \text{W, and Match(X, Y, F)} \\ \text{undefined otherwise} \end{cases}$$

It should be noted that $\text{Probe}_{\text{Local}}$ makes no use of the notions “specifier” or “head.” Indeed, it assumes no structural asymmetry at all, only the valued-unvalued asymmetry.

It should also be noted that, since so-called “spec-head” structures, especially those associated with Case and agreement, are often formed by Internal Merge, our final version of $\text{Agree}_{\text{Local}}$, much like long-distance Agree, will need to replace every instance of the object being valued/checked. Therefore, our final version of $\text{Agree}_{\text{Local}}$, is defined as in (83).

(83) Where SO is a syntactic object F is a feature type, and v is a feature value $\neq 0$ and G is a lexical item token such that $\text{Probe}_{\text{Local}}(\alpha, \text{X, F}) = \text{G}$, where SO = αor SO is contained in α,

$$\text{Agree}_{\text{Local}}(\text{SO, G, F}_\text{v}) = \begin{cases} \text{Value(SO,}\langle\text{F, v}\rangle\text{) if SO=G} & (a) \\ \text{SO if SO is a lexical item token} & (b) \\ \text{Merge(Agree(A, G, F}_\text{v}\text{), Agree(B, G, F}_\text{v}\text{)) if SO} = \{\text{A,B}\} & (c) \end{cases}$$

3.6 Summary

In this section, I provided a formal definition of one particular conception of Agree—Long-Distance Downward Valuation Agree—by first breaking it into individual pieces—Probe, Match, Value—which

I gave formal definitions, and then assembling those definitions in such a way as they define Agree. I then discussed a few alternative conceptions of Agree, showing how they could be defined by altering the previous definitions as minimally as possible. This description of the definition process might suggest that Agree is modular—that it consists of several independent operations that can be mixed and matched—but this is not the case. Rather, while the discussion of each alternative tended to focus on a single operation, the changes to that operation was such that it necessitated minor modifications to Agree as a whole. Agree, then, does seem to be real operation, albeit a rather complex one, as I will demonstrate in the next section.

4 Properties of Agree

With the Agree operation properly formalized, we are in a position to investigate the operation's theoretical properties, which have either not been remarked upon in the literature, or been discussed without the precision that formalization allows. This section will discuss some of those properties. Rather than investigating Agree in isolation and following the premise that Agree is a full-fledged derivational operation like Merge, Select and Transfer, this section will focus on those properties of Agree that distinguish it from other operations—Merge in particular.

We will first see, in section 4.1, that Agree differs from Merge and Select in that it is inherently recursively defined, while the latter two are defined non-recursively. Related to this, I will argue in section 4.2 that the fact that our definition of Agree includes instances of Merge effectively rules out any general Agree requirement for Merge. In section 4.3, I show that, unlike Merge and Select, Agree does not close the set of syntactic objects, and that attempts to rectify this leads to problematic predictions for language acquisition. Finally, in section 4.4 I discuss the implications of Agree for the NTC.

4.1 $\mathrm{UG}_{\mathrm{Agree}}$

In order to do so, though, we must give a definition of $\mathrm{UG}_{\mathrm{Agree}}$ in (84) and derivation in (85).

(84) Universal Grammar is a 7-tuple:

$\langle$PHON-F, SYN-F, SEM-F, Select, Merge, Transfer, Agree$\rangle$

(85) 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 $A \in LA_i$, $\langle LA_{i+1}, W_{i+1} \rangle =$ Select(A, $\langle LA_i, W_i \rangle$), or

(derive-by-Transfer) ...,

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

(derive-by-Agree) or $LA_i = LA_{i+1}$ and the following conditions hold for some SO, X, G and F_v:

a. $SO \in W_i$

b. SO immediately contains X

c. $\text{Probe}(SO,X,F_v) = \{G\}$

d. $W_{i+1} = (W_i - \{SO\}) \cup \{\text{Agree}(SO,G,F_v)\}$

This definition of a derivation uses the names of its procedures, but in the case of Merge and Select, one could just as easily expand them to give their full definition fully in terms of set-theory because they are non-recursive operations. Agree, however, is recursively defined, that is, it is defined in terms of itself—"Agree" appears on the left-hand and right-hand side of the equals sign in (60)—so such an expansion is not possible. This is a fundamental difference between Agree and the other generative operations—Merge and Select are non-recursive functions, while Agree is recursive.[15]

---

[15] Interestingly, C&S also define Transfer recursively. It follows then that Transfer should also be considered a different kind to operation—a conclusion also predicted by the fact that Transfer is generally considered an operation of the interfaces rather than Narrow Syntax.

Beyond its recursive definition, there are a number of properties that set Agree apart from its fellow operations. First, since performing Agree on a syntactic object entails searching the object, modifying certain constituents, and putting the object back together, and since objects can only be put together by applying Merge, every non-trivial application of Agree includes at least one application of Merge. This is reflected in definitions (60) and (75)—in which Merge appears in the intension of Agree—and concurs with Hornstein (2009, pp. 126–154) who notes that the minimal c-command relation required by Agree (Specifically non-local Agree, or AGREE in his terminology) is exactly the same as the one that is assumed to hold in all cases of Internal-Merge (which he calls "Move"). Hornstein's critique, that Agree and Internal-Merge are redundant, is actually complementary to the fact that Agree as defined entails Merge. The former suggests that either Agree or Internal Merge should be eliminated, while the latter rules out eliminating Internal-Merge.

4.2 Agree as a prerequisite for Merge

Early in the minimalist program, Chomsky (2000) proposed that Agree was a prerequisite for Move—that Move was a reflex of Agree. Merge—what we now call External Merge—on the other hand, was free to apply without Agree. Once Internal Merge was discovered, though, theorists were faced with a dilemma—if Merge and Move were truly a single operation, they couldn't very well have different prerequisites. There are two ways out of this dilemma—either all instances of Merge are free, or all instances of Merge require Agree.[16] Although C&S's formalization and my extension of it assume that all operations, except perhaps Transfer, are free, there are Agree theorists—for instance Wurmbrand (2014)—who take Agree to be a prerequisite to Merge. Therefore, in this section, I will discuss the barriers to modifying the formal grammar to make Agree a prerequisite for Merge.

The principle barrier to making Agree a prerequisite for Merge is that, as defined in (85), the derivation is a computational procedure and, therefore, is strictly incremental. That is, the validity of a given stage S_n ($n \neq 1$) depends solely on its form and the form of the immediately preceding stage S_{n-1}. Requiring every instance of Merge to be preceded by an instance of Agree, however, would mean that the validity of

---

[16] See Boeckx (2010) for a broader discussion of the schism.

a stage S_n ($n \neq 1$) depends on its two preceding stages S_{n-1} and S_{n-2}. That is, S_n can be derived from S_{n-1} by Merge only if S_{n-1} is derived from Sn-2 by Agree. A derivation, then, would need memory, albeit a very small amount of it.

On its face, this does not seem to be an insurmountable barrier, but as we shall see, it will end up ruling out the first instance of Merge in any derivation. To begin with, we reformulate our definition of derivation by adding the underlined line in our derive-by-Merge clause in (86).

(86) 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-Transfer) ...,

(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,

c. $\underline{\langle W_i, LA_i \rangle \text{ is derived by Agree from } \langle W_{i-1}, LA_{i-1} \rangle}$, and

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

(derive-by-Agree) or $LA_i = LA_{i+1}$ and the following conditions hold for some SO, X, G and F_v:

a. $SO \in W_i$

b. SO immediately contains X

c. $\text{Probe}(SO,X,F_v) = \{G\}$

d. $W_{i+1} = (W_i - \{SO\}) \cup \{\text{Agree}(SO,G,F_v)\}$

Now, lets consider an abstract subderivation of the syntactic object {X, Y} where X and Y are lexical item tokens. We start in S_1, given in (87) with an empty workspace and a lexical array containing at least X and

Y.

(87)
$$\begin{aligned} S_1 &= \langle LA_1, W_1 \rangle \\ &= \langle \{X, Y, Z \ldots\}, \{\} \rangle \end{aligned}$$

Next we perform Select twice, to bring X and Y into the workspace.

(88)
$$\begin{aligned} S_2 &= \text{Select}(X, S_1) \\ &= \langle \{Y, Z \ldots\}, \{X\} \rangle \end{aligned}$$

(89)
$$\begin{aligned} S_3 &= \text{Select}(Y, S_2) \\ &= \langle \{Z \ldots\}, \{X, Y\} \rangle \end{aligned}$$

Under a free Merge grammar, we would, at this point simply Merge X and Y, but this option is not available to us, since derive-by-Merge in (86) requires an Agree step. A Select step is possible here, but that would only postpone our dilemma. We need to perform Agree next.

Assuming that X could value Y for feature F—i.e., Match(X, Y, F) = 1—let's consider the structural prerequisites. As stated in (86), X and Y must be contained in the same syntactic object SO, which, in turn, must be a member of the workspace. In S_3, however, both X and Y are members of the workspace, and there is no SO to speak of. No stage S_4, then, can be derived by Agree.

We've arrived then at an instance of circularity—every instance of Merge requires a preceding instance of Agree, and every instance of Agree requires a preceding instance of Merge. First Merge, then, is impossible if the definition of a derivation in (86) holds.[17]

4.3 The Non-Closure of Agree

Since a computational procedure is essentially the repeated application of an operation, or set of operations, with each application providing the input for the following application, the domain of a given

---

[17] This is not to say that tying Agree to Merge in some way will always be a dead-end. On the contrary, one of, for instance, Hornstein's (2009) critiques of long-distance Agree is that it ties Agree too loosely to Merge. Merge creates the structural conditions for Agree—a point which Local Agree more or less explicitly acknowledges. This leads one to wonder why we consider Merge and Agree to be distinct operations—why Agree is not treated as a reflex of Merge. The obvious response to this is that there do seem to be instances of long-distance agreement that do not involve movement. This objection, however, only holds if we rule out the covert movement hypothesis, which states that apparent long-distance agreement relations are, in fact, cases of movement in which the lower copy of the moved element is pronounced (see Chomsky, 1993, 1995). This hypothesis has fallen out of fashion due to empirical issues such as those discussed by Hornstein (2009, pp. 135–153), but this section suggests that it may face fewer theoretical hurdles than long-distance.

computational operation must be closed under that operation, as defined in (90).

(90) Domain D is **closed under** n-place operation f iff
for all $x_0, x_1, \ldots x_n \in$ D $f(x_0, x_1, \ldots, x_n) \in$ D.

In the case of our syntactic derivations, our domain is the set of stages, which C&S demonstrate are closed under derive-by-Select and derive-by-Merge. I have thus far been assuming that it is also closed under derive-by-Agree, but that assumption is perhaps not strictly true, under our present definitions.

As defined, derive-by-Agree is a function from stages to stages that modifies a stage's workspace, by performing Agree on a syntactic object in that workspace. Therefore, the set of stages is closed under derive-by-Agree iff the set of syntactic objects is closed under Agree. For its part, Agree operates on a given syntactic object SO by applying Value to SO if SO is an appropriate lexical item token, or to the appropriate lexical item tokens contained in SO otherwise. Therefore the set of syntactic objects is closed under Agree iff the set of lexical item tokens is closed under Value. We need only consider a simple instance of Value to see that this is not obviously the case.

Consider the lexical item token X_k, defined in (91), which has only one syntactic feature, [F:0].

(91) $X_k = \langle\langle \mathrm{PHON}_X, \{\langle F, 0\rangle\}, \mathrm{SEM}_X\rangle, k\rangle$
where $\mathrm{PHON}_X \in$PHON-F*, $\mathrm{SEM}_X \subset$SEM-F, k is an integer, and $\langle F, 0\rangle \in$SYN-F.

What about the result of applying Value to X_k, given in (92)?

(92) $\mathrm{Value}(X_k, \langle F, v\rangle) = \langle\langle \mathrm{PHON}_X, \{\langle F, v\rangle\}, \mathrm{SEM}_X\rangle, k\rangle$
where v is a non-zero integer.

Since PHON_X, SEM_X, and k are unchanged, the new object is a lexical item token iff $\langle F, v\rangle \in$SYN-F. That is, the set of lexical item tokens is closed under Value only if the universal set of syntactic features in $\mathrm{UG}_{\mathrm{Agree}}$ contains both valued and unvalued features.

While there is no strictly formal reason for modifying our theory features by hypothesizing that SYN-F contains valued and unvalued features, such a hypothesis would put us in something of a theoretical quandary. In the grammar assumed by this paper, language acquisition is at least partially a process of constructing lexical items from universal feature sets so that they match tokens in the primary linguistic data. The basic premise of Agree theory, though, is that a unvalued features cannot surface and therefore must be valued during the derivation. If this is the case, then there are effectively no tokens of unvalued features in the primary linguistic data. Why, then, would a language acquirer ever construct a lexical item with an unvalued feature?

To take a concrete example, consider the case of French adjectives which show gender and number agreement as demonstrated in (93).

(93)

	Sg	Pl
Fem	*grande*	*grandes*
Masc	*grand*	*grands*

This situation is consistent with two sorts of lexicons if we assume lexically valued SYN features—lexicons with multiple *adj* LIs, each with valued φ-features as in (94) and lexicons with a single *adj* LI with unvalued φ-features as in (95).

$$(94)\quad \text{LEX} = \left\{ \begin{array}{c} \ldots, \\ \langle \textit{/-e/}, \{\langle\gamma,1\rangle, \langle\#,1\rangle\}, \text{SEM}_{adj}\rangle, \\ \langle \textit{/-es/}, \{\langle\gamma,1\rangle, \langle\#,2\rangle\}, \text{SEM}_{adj}\rangle, \\ \langle \emptyset, \{\langle\gamma,2\rangle, \langle\#,1\rangle\}, \text{SEM}_{adj}\rangle, \\ \langle \textit{/-s/}, \{\langle\gamma,2\rangle, \langle\#,2\rangle\}, \text{SEM}_{adj}\rangle, \\ \ldots \end{array} \right\}$$

$$(95)\quad \text{LEX} = \{\ldots, \langle \text{PHON}_{adj}, \{\langle\gamma,0\rangle, \langle\#,0\rangle\}, \text{SEM}_{adj}\rangle, \ldots\}$$

Since the lexicon in (94) represents a surface analysis of adjective morphology, it would be the more straightforward to acquire than (95) which requires an additional step of abstraction from the data. All else being equal, then, allowing SYN-F to contain valued features would seem to predict the sort of lexicon in (94) for French. This, of course would be consistent with a checking-based Agree, but not a valuation-based Agree.

Alternatively, we could assume that all and only unvalued features are members of SYN-F—stated formally as an axiom in (96).

$$(96)\quad \text{For all features } \langle \text{F}, \text{v}\rangle, \text{v} = 0 \leftrightarrow \langle \text{F}, \text{v}\rangle \in \text{SYN-F}$$

This would remove the acquisition issue—(94) would be an impossible lexicon—and would be consistent with the basic premise of Agree theory. It still would require theoretical explanation, but of the more general sort suggested in footnote 10. but it would mean that the set of lexical item tokens is not closed under Value, and therefore the set of stages is not closed under a Value-based Agree. A Value-based Agree, then, could not be a computational operation in a version of $\mathrm{UG}_{\mathrm{Agree}}$ with (96) as an axiom.

In sum, in order for a valuation-based Agree such as the one defined in (60) to be a viable as a computational procedure, we must expand the domain of possible lexical items in a theoretically questionable way.

4.4 Agree and the NTC

One of the theorems of C&S's formal grammar is the No Tampering Condition defined by Chomsky (2007, p. 8) as follows: "Suppose X and Y are merged. Evidently,efficient computation will leave X and Y unchanged (the No-Tampering Condition NTC). We therefore assume that NTC holds unless empirical evidence requires a departure from [the strong minimalist thesis] in this regard, hence increasing the complexity of UG." C&S's formulation of NTC, which they prove as a theorem of UG, is given in (97).

(97) For any two consecutive stages in a derivation $S_1 = \langle LA_1, W_1\rangle$ and $S_2 = \langle LA_2, W_2\rangle$,
for all A contained in W_1, A is contained in W_2.

Since the effect of every form of Agree defined in this paper is to replace all instances of some lexical item token G in a workspace with a distinct item G', Agree violates NTC by design. The issues $\mathrm{UG}_{\mathrm{Agree}}$ discussed above, then, may be predicted by Chomsky's conjecture that UG operations conform to the NTC. There are essentially two ways of dealing with this result—either we take the approach that C&S take with Transfer and modify Agree so that it does not violate NTC, or we argue that "empirical evidence requires a departure from" NTC. I will discuss each of these options in turn below.

4.4.1 NTC-Respecting Agree

A straightforward way of constructing an Agree operation that respects the NTC is to formally separate the content of a derived expression from its structure in some way with Merge manipulating the structure and Agree manipulating the content. A stage of the derivation, then would consist of a lexical array, a workspace, and ledger as in the definition in (98)

(98) A stage is a triple S = ⟨LA, W, L⟩, where LA is a lexical array, W is a set of syntactic objects, and L is a set of pairs of lexical item tokens. We call W the workspace of S and L the ledger of S.

Rather than modifying lexical item tokens in place, Agree would add a pair $\langle \text{LI}_k, \text{LI}'_k \rangle$, where LI_k is a lexical item token contained in the workspace and LI'_k is the result of Valuing LI_k for some feature. The ledger, then, postpones the tampering of Agree, either until Transfer, or until the SM and/or the CI system and thereby rescues the NTC.

This sort of move also fixes a number of issues already discussed regarding Agree. A version of Agree that respects NTC does not alter the workspace—it merely constructs an ordered pair and adds it to the ledger. It does not take apart and put back together an already constructed syntactic object, as standard Agree as defined in (60) does. Therefore it does not need to be recursively defined, and it does not need to refer to Merge in its definition.

This improvement aside, however, it also lays bare the fact that Agree as a syntactic-derivational operation is fundamentally redundant. The prerequisites for Agree are a structural relation (Search) and content relation (Match) between two lexical item tokens. So, suppose X and G are lexical item tokens and, for some feature F, Match(X,G,F)=1. Further suppose that stage S_n in derivation D is derived by Merge(X, Y), where Y contains G and no lexical item token H, such that Match(X,H,F)=1. At this point, our prerequisites are met and we can perform Agree, but supposing instead we derive stages S_{n+1} and S_{n+2} by Selecting and Merging another lexical item token. By the NTC, the object {X, Y} is contained in the root object of S_{n+2}, and therefore all of the structural and content relations that held at S_n still hold at S_{n+2} including the prerequisites for X to Agree with G for F.[18] By extension, we can continue to postpone Agree at least until the next instance of Transfer without losing the prerequisites for Agree. It seems, then, that, while we can certainly define Agree so that it respects NTC, if we have NTC, we can define Agree as an interface operation, perhaps as part of Transfer. This formalization, then, represents a sharp departure from the various theories of Agree whose formalization is the task at hand, and which

---

[18] See theorems 2 and 3 in Collins and Stabler (2016).

share the assumption that Agree modifies already constructed SOs mid-derivation.

4.4.2 Agree instead of the NTC?

Even as stated by Chomsky (2007), the NTC is not an absolute law akin, say, to the law of non-contradiction. Rather, he proposes that we assume the NTC "unless empirical evidence requires a departure from [the strong minimalist thesis] in this regard." In one sense, this is a very low bar, since NTC is a universal statement, which only requires a single counterexample to invalidate. In practice though, it is far from obvious what sort of evidence would count as counterexample.

The relative ubiquity of morphological agreement, for instance, might seem to be the sort of evidence we need, but it is not sufficient to invalidate NTC. Consider, as a parallel, linear order. It is a plain fact that external linguistic expressions have linear order, yet that linear order is still assumed to be absent in the grammar—at least in standard Merge-based grammars. Yet, as Chomsky (2020) citing McCawley (1968) points out, adverbs like *respectively*, which depend on linear order for their interpretation, provide evidence that conjunction structures have inherent linear order.

(99) Beth and Sara met Hanako and Máire respectively.
  a. = Beth met Hanako and Sara met Máire.
  b. ≠ Beth met Máire and Sara met Hanako.

What we need, then, is evidence that standard Agree is occurring in a derivation interspersed with Merge. Preminger (2014) argues that we have exactly such evidence in the interrelation of morphological case, φ-agreement, and subject position.[19] The form of the argument is given in (100)

(100) a. Morphological case feeds φ-agreement in quirky-subject languages.
  b. Φ-agreement feeds movement to canonical subject in non-quirky-subject languages.
  c. The functioning of the grammar is uniform across languages (The Uniformity Principle).
  d. **Therefore**, morphological case and φ-agreement precede movement to subject.

---

[19] An anonymous reviewer points out that the evidence that Richards (2001) adduces for "tucking-in" is perhaps stronger evidence of a violation of NTC. I address Preminger's argument here because it has to do directly with agreement and is therefore germane to the topic at hand. See Hornstein (2009), though, for a proposal that predicts the effects of tucking-in without tucking-in.

e. **Therefore**, morphological case and φ-agreement are part of the narrow syntax.

The argument is logically sound, but it depends on an analysis of the evidence that is plausible, but not the only possible analysis. That is, it depends of the truth of the first two premises, which are empirical statements. Despite being empirical statements, though, they depend on two theoretical notions—"quirky subjects" and "canonical subject position"—to even be coherent. I will take for granted that the term "quirky subject" is coherent, and focus on "canonical subject position."[20]

Furthermore, it is worth noting, that Preminger frames his premises in terms of "feeding" rather than "driving" or "triggering." An operation X feeds another operation Y if X creates the necessary conditions for Y and X precedes Y. "Feeding", then, speaks to the order of operations more than causation.

One property of canonical subject position that Preminger is clear about is that it is syntactic—he says of movement to canonical subject position that it is "clearly syntactic (since it creates new binding configurations, for example)" (p177) and that it "is a syntactic process par excellence" (p184). We further know, based on the second premise of (100), which Preminger claims as an empirical result, that movement to canonical subject position in non-quirky-subject languages should always co-occur with φ-agreement. Since this latter requirement is an empirical claim, though, it should not be too directly tied to our definition lest our reasoning be circular. We can construct our definition by applying these two desiderata to some representative data.

Our representative data is given in (101), where the underlined subexpression is could be or has been considered to be in subject position in English.

(101) a. The city is bustling.
b. There seem to be unicorns in my house.
c. The dog running down the street was quite a sight.
d. They seemed *t* to leave.
e. I expect *t*/PRO to leave shortly.

---

[20] It should be noted that the modifiers "quirky" and "canonical" both subjective in nature—they denote degrees of conformity to some norm—suggesting that the phenomena that they refer to have not yet been given a theoretical explanation, just as the terms "*Exceptional* Case Marking" and the "*Extended* Projection Principle" indicated problematic data—explananda, rather than explanantia (Chomsky, 2013, p. 35).

f. We believed them to be a capable team.

I believe that it is quite safe to label *the city* in (101a) as being in canonical subject position[21]—it is the specifier of TP and it triggers φ-agreement on the finite auxiliary. On the other hand, the existential associate *unicorns* in (101b) is likely not in a canonical subject position.[22] In fact, existential associates not being in canonical subject position gives force to the second premise of (100)—in order for φ-agreement to feed movement to canonical subject position, agreement must be necessary but not sufficient for movement and existential clauses show this only if we assume that their associates are not (possibly covertly) in canonical subject position.[23]

This leaves us with non-finite subject position in (101c) to (101f). In each of these cases, the underlined expression could reasonably be said to be in a subject position, and to have moved there, yet there is no apparent φ-agreement associated with that move. We could reasonably reject *the dog* in (101c) as being in canonical subject position, since it is not a specifier to a TP, leaving us with the null subjects in (101d) to (101e) and the ECM subject in (101f). In a summarizing table, though, Preminger (2014, p. 164) seems to assert that, in English, only nominatives are candidates for movement to canonical subject. This would rule out traces/PRO and ECM subjects as canonical subjects.

Canonical subject position, then, seems to refer to the specifier of finite T, at least in English. Assuming such a position can be defined well enough to support generalizations such as Preminger's premises,[24] the Uniformity Principle—Preminger's third premise—demands that we treat movement to the specifier of finite T either as a special case of Merge, distinct from external or ordinary internal Merge, or as derivational operation of its own, distinct from Select, Merge, and Agree. So, if we keep strictly to the theory assumed in this paper, where $\mathrm{UG}_{\mathrm{Agree}}$ has Merge, Select, Agree, and Transfer, Preminger's argument does

---

[21] We might call it the canonical canonical subject.

[22] See Hornstein (2009, pp. 130–134), though, for discussion to the contrary.

[23] The expletive *there* in (101b) seems to be in canonical subject position—if *unicorns* was there it would certainly be in canonical subject position—but it does not trigger φ-agreement. This, however, does not contradict (100b), which links φ-agreement with movement to canonical subject position, not to the position itself, if we assume that expletives are inserted in canonical subject position, not moved there.

[24] Chomsky (2013), for instance, argues that "specifier" is not definable in a theory based on simplest Merge, such as the one assumed in this paper. This is not strictly true but, whereas "specifier" was trivially definable in a system like X-Bar, which takes labelling as a primitive, any definition of "specifier" in the present system would likely consist of the coordination of multiple predicates.

not go through because the premise (100b) would not be well-defined.[25] Put another way, (100) might be coherent in some theory of grammar, but it is not coherent in a theory that assumes UG as defined in (8) or UG_{Agree} as defined in (84).

We could try to rescue (100) by restating (100b) as (100b′) which is coherent in UG_{Agree}—assuming "quirky subject" can be defined and the problems outlined in section 4.2 can be overcome.

(100b′) Φ-agreement feeds Internal Merge in non-quirky-subject languages.

In order for this new premise to be true, though, movement to non-canonical subject position must also require φ-agreement, which implies some sort of abstract or covert φ-agreement on non-finite predicates such as those in (101c) to (101f). In light of this implication, it is difficult to see how this new premise could be justified empirically, and therefore it should be rejected, or at best treated as a hypothesis. Since any argument is only as strong as its premises, this would weaken Preminger's argument a great deal.

To recap, Preminger's argument as given in (100), while seemingly logically sound, rests on the assumption that movement to canonical subject position is a bona fide syntactic operation, distinct from other types of movement. This assumption would be a departure from the theory assumed here, which takes all movement operations to be instances of Merge. Preminger's conclusion, that agreement takes place in the syntax taken with my argument above that Agree violates the NTC, implies the conclusion that the NTC should be at least weakened[26]—another departure from the theory. It would seem, then, that one departure from theory begets other departures—a result that is far from surprising and, in fact, indicates the internal unity of the theory of grammar assumed here. More importantly, Preminger's argument, the most explicitly fleshed out empirical argument in favour of Agree as a syntactic operation, should not be taken as a falsification of NTC or SMT.

---

[25] It might be argued that the theory assumed here cannot account for the range of data that Preminger discusses and should, therefore, be rejected. Such an objection, I would argue, mistakes entirely the nature of scientific, and more broadly rational, inquiry. While a full airing of this argument is beyond the scope of this paper, I will merely ask the reader to consider two points:

1. No scientific theory is or has ever enjoyed complete empirical coverage, even within its own domain.
2. Despite common narratives to the contrary, progress in the sciences is almost always led by theoretical progress rather than the collection of novel data.

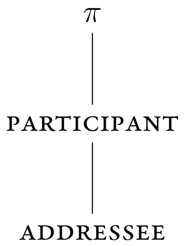
[26] Preminger (2018) builds on these results to argue against the SMT. If we do not accept his 2014 argument, we do not have to accept his later argument that depends on it,

5 Modularity and the paths not taken

Throughout the exercise in formalization, many choices were made that could have been made differently with various levels of consequence for the overall system. For instance, the choice, adopted from C&S, to include PHON as part of the LI was essentially the choice of an "early-insertion" theory of morphology. This choice, however, was of little consequence for the formalization of Agree, since it dealt exclusively with the SYN and SEM features of LIs. The choice to formalize features as type-value pairs, though, does have relevant consequences.

Suppose, for instance, I had adopted a geometric feature theory such as the one developed by Béjar (2003), where, 2nd person feature is represented as in (102).

(102)



One formal definition of feature that would capture this is given in (103), with 2nd person feature formalized as in (104)

(103) $$\text{X is a } \textit{feature} \text{ iff } \begin{cases} \text{X} \in \text{SEM} & \text{(An atomic feature)} \\ \text{or X is a pair of features.} & \text{(A complex feature)} \end{cases}$$

(104) $\langle \pi, \langle \textsc{participant}, \textsc{addressee} \rangle \rangle$

Where $\{\pi, \textsc{participant}, \textsc{addressee}\} \subset \text{SEM}$

Quite obviously, this would require us to redefine or replace our auxiliary notions like *feature-of* or *unvalued feature* and to define new ones like *depends-on* or *entails*, but most importantly it would require new definitions of Match and Value. Béjar (2003) discusses various parameters that would determine these definitions—for instance, whether *unvalued* features should be fully specified or underspecified—so I will direct readers to that discussion should they wish to formalize Match and Value under this theory of features.

On the other hand, I see no reason to expect that we must alter our Minimal Search algorithm in (55) nor our final definition of Agree in (60)to account for alternative theories of features. Minimal Search is a general purpose algorithm—it doesn't depend on the particular search criterion—and Agree searches a structure and replaces Matching lexical item tokens with the result of Value—as long as Match is a predicate that compares lexical item tokens relative to features, and Value is a function from somehow-defective lexical item tokens to less-defective lexical item tokens.

Likewise, were one able to adequately define a minimal DFS algorithm or if one adopted a ledger-based model of Agree, there would not necessarily be any reason to abandon either the type-value or the geometric theory of features. Agree, Search, and Match/Value, then are to a certain extent modular with respect to each other and, while the limits of that modularity are a purely theoretical question, the final choice of individual theories will depend on a combination of theoretical and empirical concerns.

6 Concluding remarks

The task of formalizing a theoretical conjecture occupies an odd place in the sciences. While it does generally not bring anything new to the table, it does give us the opportunity to objectively assess the validity and theoretical prospects of various informal proposals. By formalizing various proposals for Agree as a syntactic operation, we can see that what often is shown as a simple curved arrow on tree diagrams is actually a rather complicated computational operation. Not only is this complexity apparent simply from the size of the formal definition compared, say, to that of Merge, but it is reflected in the theoretical complexities identified in section 4.

In its current state, then, Agree should not be taken for granted, even with what seems to be overwhelming evidence of its existence. This, however, leaves the theory in an awkward position—the phenomena that Agree is supposed to explain appear to be real and rather ubiquitous, but our tool for explaining them is not yet ready. If we are engaged in rational inquiry (*i.e.*, science) then we should not be surprised to find ourselves in such a position. It does not mean that its time to throw up our hands and discard our current theory. It means that we have plenty of work left—an enviable position to be in.

References

Béjar, S. (2003). *Phi-syntax: A theory of agreement* (Doctoral dissertation). University of Toronto.

Béjar, S., & Rezac, M. (2009). Cyclic agree. *Linguistic Inquiry*, *40*(1), 35–73.

Bjorkman, B., & Zeijlstra, H. (2014). Upward agree is superior.

Boeckx, C. (2010). Reflections on the plausibility of crash-proof syntax, and its free-merge alternative. In M. T. Putnam (Ed.), *Exploring crash-proof grammars* (pp. 105–124). John Benjamins Publishing Company.

Branan, K., & Erlewine, M. Y. (forthcoming). *Locality and (minimal) search*. https://ling.auf.net/lingbuzz/005791

Chametzky, R. (1996). *A theory of phrase markers and the extended base*. SUNY Press.

Chomsky, N. (1957). *Syntactic structures*. Mouton.

Chomsky, N. (1965). *Aspects of the theory of syntax*. MIT Press.

Chomsky, N. (1993). A minimalist program for linguistic theory. In K. Hale & S. J. Keyser (Eds.), *The view from building 20: Essays in linguistics in honor of sylvain bromberger*. MIT press.

Chomsky, N. (1995). *The minimalist program*.

Chomsky, N. (2000). Minimalist inquiries: The framework. *Step by step: Essays on minimalist syntax in honor of Howard Lasnik*, 89–155.

Chomsky, N. (2004). Beyond explanatory adequacy. In A. Belletti (Ed.), *Structures and beyond* (pp. 104–131). Oxford University Press.

Chomsky, N. (2007). Approaching ug from below. In U. Sauerland & H.-M. Gärtner (Eds.), *Interfaces + recursion = language? chomsky's minimalism and the view from syntax-semantics* (pp. 1–29). Mouton de Gruyter Berlin.

Chomsky, N. (2013). Problems of projection. *Lingua*, *130*, 33–49.

Chomsky, N. (2020). *The ucla lectures*. https://ling.auf.net/lingbuzz/005485

Church, A. (1941). *The calculi of lambda-conversion*. Princeton University Press.

Collins, C., & Groat, E. (2018). *Copies and repetitions*. https://ling.auf.net/lingbuzz/003809

Collins, C., & Stabler, E. (2016). A formalization of minimalist syntax. *Syntax*, *19*(1), 43–78. https://doi.org/10.1111/synt.12117

Ermolaeva, M. (2018). Morphological agreement in minimalist grammars. In A. Foret, R. Muskens, & S. Pogodalla (Eds.), *Formal grammar* (pp. 20–36). Springer Berlin Heidelberg.

Halle, M., & Marantz, A. (1993). Distributed morphology and the pieces of inflection. *The view from building 20* (pp. 111–176). The MIT Press.

Harbour, D. (2007). *Morphosemantic number: From Kiowa noun classes to UG number features*. Springer.

Harley, H., & Ritter, E. (2002). Person and number in pronouns: A feature-geometric analysis. *Language*, *78*(3), 482–526.

Harris, Z. S. (2002). The background of transformational and metalanguage analysis. In B. Nevin (Ed.), *The legacy of zellig harris: Language and information into the 21st century. Vol. 1. Philosophy of science, syntax and semantics* (pp. 1–18). J. Benjamins Pub. Co.

Hornstein, N. (2009). *A theory of syntax: Minimal operations and universal grammar*. Cambridge University Press.

Ke, H. (2019). *The syntax, semantics and processing of agreement and binding grammatical illusions* (Doctoral dissertation). University of Michigan.

McCawley, J. D. (1968). The role of semantics in a grammar. In E. Bach & R. Harms (Eds.), *Universals in linguistic theory* (pp. 124–169). Holt, Rinehart & Winston.

Partee, B. B., ter Meulen, A., & Wall, R. (1990). *Mathematical methods in linguistics*. Kluwer Academic Publishers.

Preminger, O. (2013). That's not how you agree: A reply to zeijlstra. *The Linguistic Review*, *30*(3), 491–500.

Preminger, O. (2014). *Agreement and its failures* (Vol. 68). MIT press.

Preminger, O. (2018). Back to the future: Non-generation, filtration, and the heartbreak of interface-driven minimalism. *Syntactic structures after 60 years: The impact of the chomskyan revolution in linguistics* (pp. 355–380). De Gruyter.

Preminger, O. (2019). What the pcc tells us about “abstract” agreement, head movement, and locality. *Glossa: A Journal of General Linguistics*, *4*(1), 13. https://doi.org/10.5334/gjgl.315

Pullum, G. K. (2011). On the mathematical foundations of syntactic structures. *Journal of logic, language and information*, *20*(3), 277–296.

Richards, N. (2001). *Movement in language: Interactions and architectures*. Oxford University Press.

Rooryck, J., & Wyngaerd, G. V. (2011). *Dissolving binding theory* (Vol. 32). OUP Oxford.

Stabler, E. (1997). Derivational minimalism. *Logical Aspects of Computational Linguistics: First International Conference, LACL’96, Nancy, France, September 23-25, 1996. Selected Papers*, *1328*, 68.

Starke, M. (2010). Nanosyntax: A short primer to a new approach to language. *Nordlyd*, *36*(1), 1–6.

Wurmbrand, S. (2014). The merge condition : A syntactic approach to selection. In P. Kosta, S. Franks, T. Radeva-Bork, & L. Schürcks (Eds.), *Minimalism and beyond: Radicalizing the interfaces* (pp. 130–166). John Benjamins Publishing Company.

Zeijlstra, H. (2012). There is only one way to agree. *The linguistic review*, *29*(3), 491–539.