A formalization of Agree as a derivational operation

(Formerly: "Agree as derivational operation: Its definition and discontents")

Daniel Milway

30th July 2021

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 a version of long-distance Agree in which a higher object values a feature on a lower object, and modify that definition to reflect various several versions of Agree that have been proposed in the "minimalist" literature. I then discuss the theoretical implications of these formal definitions, arguing that Agree (*i*) muddies our understanding of the evolution of language, (*ii*) requires a new conception of the lexicon, (*iii*) objectively and significantly increases the complexity of syntactic derivations, and (*iv*) unjustifiably violates NTC in all its non-vacuous forms.

Keywords: theory, formalization, minimalism, agree, derivations

1 Introduction

Being computational theories of grammar, minimalist Principles & Parameters theories deal mainly in procedures which generate linguistic expressions from atoms in an incremental fashion. That is, these theories traffic in computational procedures that relate stage n of a derivation to stage $n + 1$ 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 twin goals of (a) ensuring that for an arbitrary derivation stage, any application of Merge would have a single predictable result, even if that result is failure, while (b) maintaining its descriptive adequacy. Much of the current literature in minimalist P&P grammar, however, assumes the existence of a second core procedure, Agree, which, I argue in this paper, has yet to be sufficiently defined as a computational procedure.

The correct characterization of Agree ultimately depends on empirical and theoretical considerations and, while virtually the entire contemporary Agree literature focuses on the former to the exclusion of the latter, this paper seeks to contribute to the latter.[1] 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 and Rezac 2009), Local Agree (Hornstein 2009), Fallible Agree (Preminger 2014), and Upward Agree (Bjorkman and 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.

This proliferation of theories of Agree is further exacerbated by the fact that, since its inception, Generative Grammar has always had both derivational and representational expressions. In the theory used in *Aspects* (Chomsky 1965), for instance, (1) can be given three

[1] "Theory" and its derived terms are widely misunderstood within contemporary syntactic research. I take "theory" to refer to a logical system which is hypothesized to explain some domain of nature, and "theoretical" work to refer to work that investigates the internal logical properties of a theory. The work that is taken to fall under the umbrella of "theoretical syntax," however, is more often than not data analysis work—*i.e.*, empirical work—which (a) does not involve quantitative analysis—as opposed to "corpus work"—and (b) ignores the method of gathering the analyzed data—to differentiate it from "experimental work" and "field work." See Chametzky (1996) for related discussion.

formal expressions—one derivational expression in (2), and two representational expressions in (3) and (4).

(1) Sincerity may frighten the boy.

(2) a. S → NP⌒Aux⌒VP (Chomsky 1965, p. 68)

VP→V⌒NP

NP→Det⌒N

NP→N

Det→*the*

Aux→M

b. M→*may*

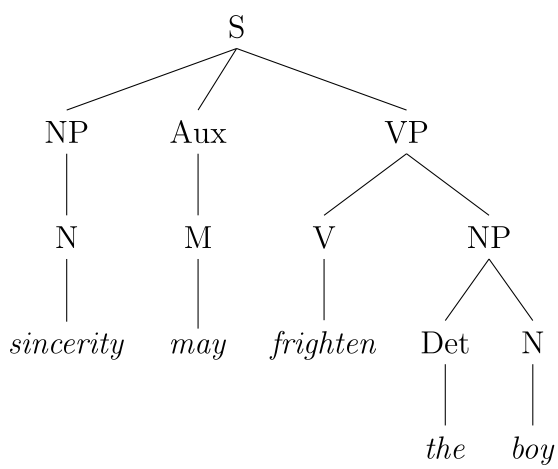
N→*sincerity*

N→*boy*

V→*frighten*

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

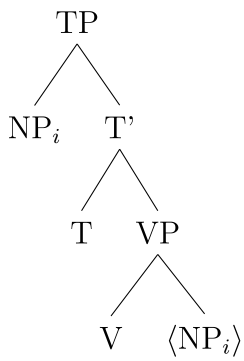
(4)



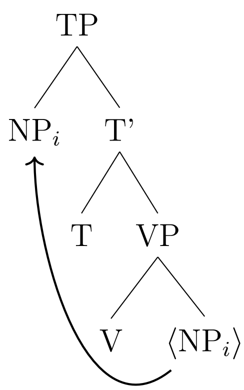
Since Generative Grammar is a computational theory, the derivational expression of a given analysis has always been the ultimate expression. The representational expressions, on the other hand, are much more concise and accessible, so they have been overwhelmingly used as shorthands for the derivational expressions, but they are useful as short-hands only insofar as they are isomorphic 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)

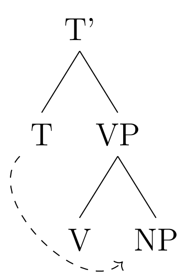


(6)



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 formal grammar developed by Collins and Stabler (2016). I sketch out this grammar, 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 computational complexity, 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: $\langle$PHON-F, SYN-F, SEM-F, Select, Merge, Transfer$\rangle$

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).[2]

(9) A lexical item is a triple: LI = $\langle$PHON, SYN, SEM$\rangle$
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 $\langle$Lex, UG$\rangle$, 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.[3]

(12) A lexical item token is a pair: $\mathrm{LI}_k = \langle \mathrm{LI, k} \rangle$, 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.

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

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

(17) A stage is a pair $S = \langle LA, W \rangle$, where LA is a lexical array 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, 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—C&S devote 5 sections of their paper to developing its definition—and, quite frankly, irrelevant to our discussion here. I will therefore omit the definition of Transfer from this paper.

(18) Given any two distinct syntactic objects A, B, 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 Select(LA, S) = $\langle LA - \{A\}, W \cup \{A\} \rangle$

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

(20) 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∈LA i , $\langle LA_{i+1}, W_{i+1} \rangle$ = Select(A, $\langle LA_i, W_i \rangle$), or
(derive-by-Transfer) . . . , or
(derive-by-Merge) $LA_i = LA_{i+1}$, and the following conditions hold for some A,B:

a. $A \in W_i$

b. Either A contains B or W_i immediately contains B, and

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

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 or Search—(*b*) the substantive prerequisite—Match—and (*c*) the process of modifying the 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 objects, rather the Probe and Goal, as they are commonly called, stand in a c-command-plus-relativized-minimality relation as specified in (21).

(21) A Probe P and Goal G can Agree iff, P c-commands G, G Matches P, and there is no head H such that H Matches P, P c-commands H and H c-commands G.

LDDV-Agree is downward in the sense that it modifies the c-commanded Goal, and it is valuation-based in the sense that the Goal is modified by converting one of its unvalued feature into a valued one as specified in (22) and (23).

(22) A Goal G Matches a Probe P for feature F iff P has [F:*val*] and G has [F:__]

(23) If P and G Agree for feature F then [F:__] on G 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). 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. We can define features, then, as in (24).[4]

(24) A *feature* is a pair $\langle \mathrm{F}, \mathrm{v} \rangle$, where v is an integer. F is called the *feature type*, v is the *feature value*.

(25) For all feature types F, $\langle \mathrm{F}, 0 \rangle$ is an *unvalued* F feature.

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

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

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 (28).

(28) For any two lexical item tokens P, G feature type F,
Match(P, G, F) = 1 iff for feature value $\mathrm{v} \neq 0$ $\langle \mathrm{F}, \mathrm{v} \rangle$ is a feature of P and $\langle \mathrm{F}, 0 \rangle$ is a feature of G.

Value is essentially a replacement operation—operating on a lexical item token, swapping an unvalued feature with a valued counterpart. This is defined in (29).

(29) For lexical item token $\mathrm{LI}_k = \langle\langle \mathrm{SEM}, \mathrm{SYN}, \mathrm{PHON} \rangle, \mathrm{k} \rangle$, and feature $\langle \mathrm{F}, \mathrm{v} \rangle$,
$\mathrm{Value}(\mathrm{LI}_k, \langle \mathrm{F}, \mathrm{v} \rangle) = \langle\langle \mathrm{SEM}, (\mathrm{SYN} - \{\langle \mathrm{F}, 0 \rangle\}) \cup \{\langle \mathrm{F}, \mathrm{v} \rangle\}, \mathrm{PHON} \rangle\ \mathrm{k} \rangle$

[4] An anonymous reviewer points out that features are more commonly assumed to be organized into hierarchical feature geometries (Béjar 2003; Harbour 2007; Harley and Ritter 2002). In section 5 discuss the formalization of one such feature theory and its limited effect on the overall formal definition of Agree.

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

3.1 Minimal Search

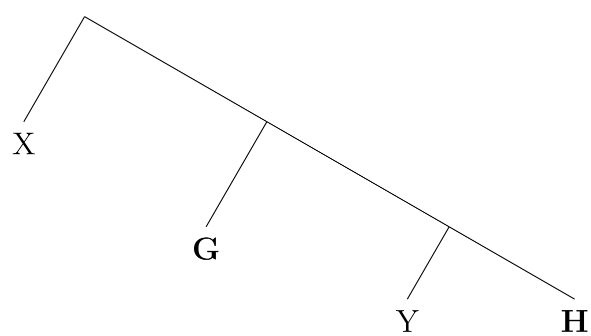
The term Minimal Search, as its usually used in minimalist syntactic theory, refers to an algorithm that retrieves the "highest" object in a structure that meets some particular criterion. In the case of Probe, that criterion is Match as defined in (28). In order to properly define such an algorithm 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 (30) is the most straightforward—G asymmetrically c-commands H, so Minimal Search retrieves G and not H.

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

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

b.

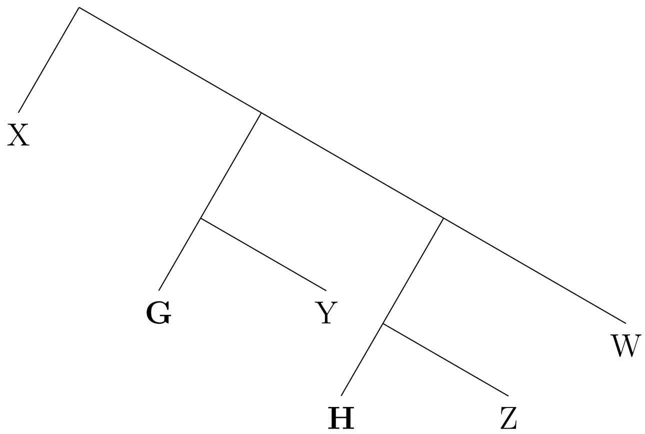


The second case in (31) is slightly more complicated—G does not c-command H, but Minimal Search should retrieve G because it is immediately contained in an object that asymmetrically c-commands H.

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

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

b.

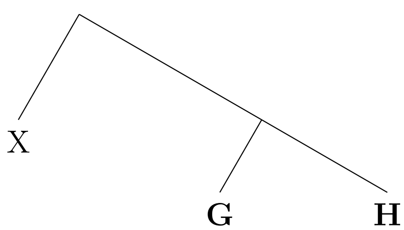


Other cases, though, will give ambiguous results. These are cases in which G and H are equidistant from the root. In (32), for instance G and H are siblings, while in (33) they are immediately contained, respectively, by siblings.

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

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

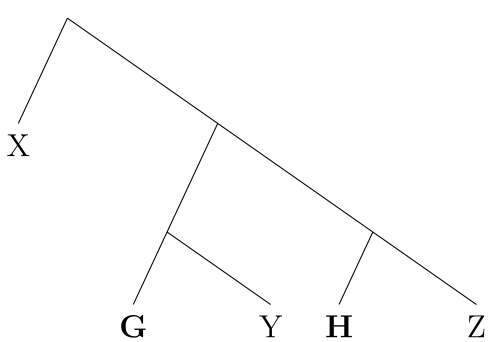
b.



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

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

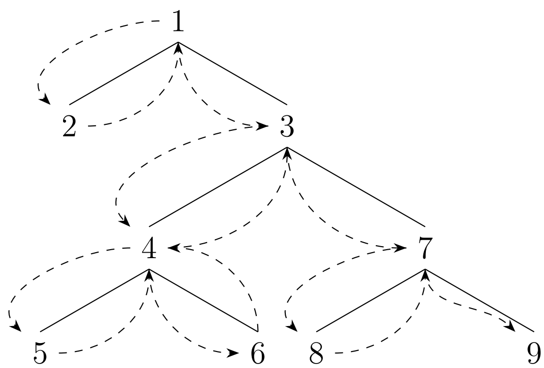
b.



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

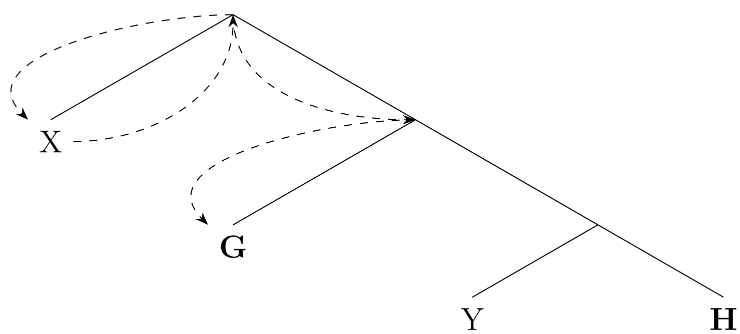
(DFS) and Breadth-First Search (BFS). DFS starts at the root of an object and searches to a terminal node before backtracking, as represented in (34), where the arrows and the numbers indicated the search order.

(34)



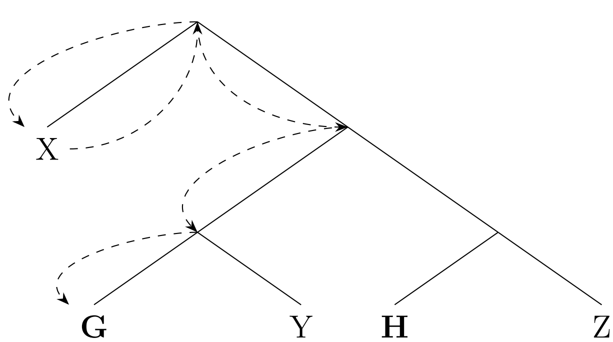
A DFS algorithm can be made minimal by designing it to stop as soon as it finds a node that meets its criterion. So, a Minimal DFS on Case 1 would be proceed as in (35) selecting.

(35)



However in an ambiguous case, like Case 4, a Minimal DFS will incorrectly retrieve just a single object as shown in (36).

(36)

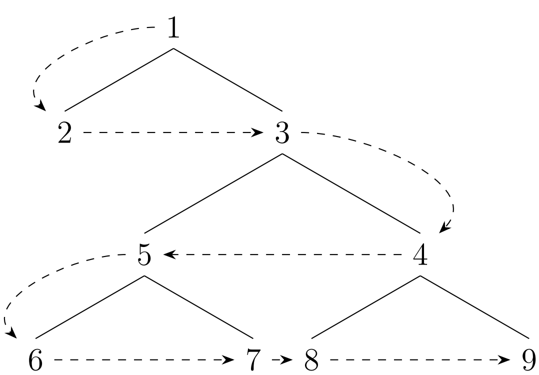


A Minimal DFS algorithm, then is over-definite—it gives a definite result where we expect an ambiguous one.

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 (35) and (36), 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.

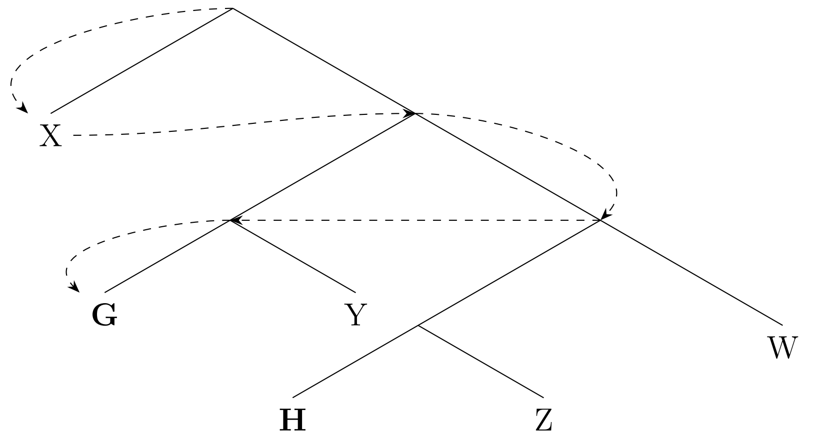
Breadth-first Search (BFS) algorithms, on the other hand, searches neighbour nodes before proceeding lower in the tree as represented in (37), where the arrows and the numbers indicated the search order.

(37)



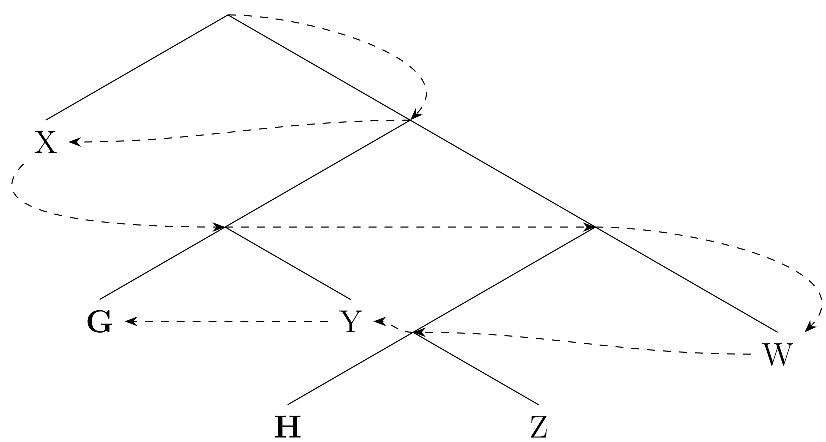
Again, this can be made minimal by requiring that the algorithm stop immediately upon finding an object that matches the search criterion. A Minimal BFS on Case 2, then, is represented in (38).

(38)



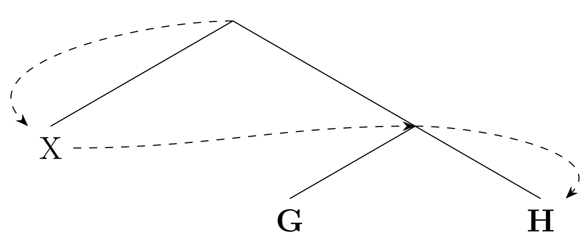
Like the Minimal DFS, the Minimal BFS, as represented in (37) and (38) assumes that nodes are linearly ordered, even if that order is arbitrary. Unlike the Minimal DFS, the order of the neighbour nodes does not matter, at least for definite cases like Case 1 and Case 2. To demonstrate this, consider the reverse version of (38) in (39).

(39)

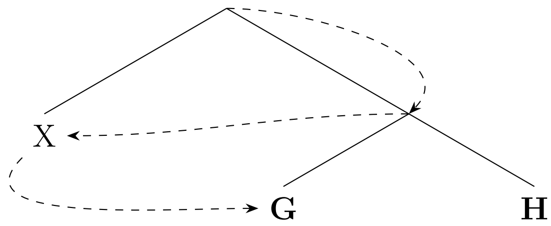


In an ambiguous case, though, Minimal BFS suffers the same fate as Minimal DFS—it is over-definite. So, in Case 3, Minimal BFS will wrongly retrieve either G or H depending on the ordering of nodes, as shown in (40) and (41).

(40)

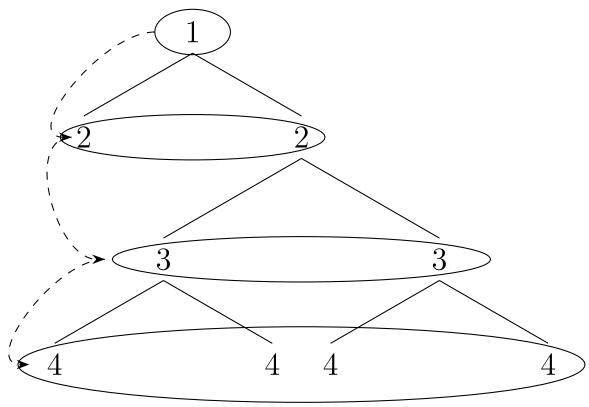


(41)



This flaw, however, can be overcome if, instead of traversing each node, we treat the sets of neighbour nodes as tiers, as in (42).

(42)



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 (43), an ambiguous search result as in (44), and a failed search as in (45).

(43) For a syntactic object SO and criterion P, Search(SO,P) is definite iff |Search(SO,P)|=1

(44) For a syntactic object SO and criterion P, Search(SO,P) is ambiguous iff |Search(SO,P)| > 1

(45) For a syntactic object SO and criterion P, Search(SO,P) is failed iff 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 (31) are given in (46)

(46) Tier 1 $= \{\mathrm{X}, \{\{\mathbf{G}, \mathrm{Y}\}, \{\{\mathbf{H}, \mathrm{Z}\}, \mathrm{W}\}\}$

$$\text{Tier 2} = \begin{Bmatrix} \{\mathbf{G}, \text{Y}\}, \\ \{\{\mathbf{H}, \text{Z}\}, \text{W}\} \end{Bmatrix}$$

$$\text{Tier 3} = \begin{Bmatrix} \mathbf{G}, \text{Y}, \\ \{\mathbf{H}, \text{Z}\}, \\ \text{W} \end{Bmatrix}$$

$$\text{Tier 4} = \{\mathbf{H}, \text{Z}\}$$

$$\text{Tier 5} = \{\}$$

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 (47).

(47) For a set X=$\{x_0, \ldots, x_n\}$ the arbitrary union of X, $\bigcup$X=$x_0 \cup \cdots \cup x_n$.

Therefore we can define a procedure NextTier in (48) and with it, Search in (49).

(48) For T, a set of syntactic objects, NextTier(T)= $\bigcup${SO∈T: SO is not a lexical item token}.

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

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

Probe, then is a special type of Search, where the search criterion is based on Match, is shown in (50).

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

Probe(SO,P,F) = Search(SO, (λx) (Match(P, x, F)))

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.

3.1.1 The appeal of DFS and attempts to rescue it

Although DFS is emprically/descriptively inadequate—given the theoretical assumptions of this paper—it retains a certain theoretical and aesthetic appeal. This appeal may come from the fact that it can be given a simple recursive definition using only the primitive concepts such as set-membership. In contrast, BFS as defined in (49) requires the definition of the ad hoc notion of a tier, which I have implicitly defined in (48) using an arbitrary union—a function whose computational definition is likely more complex than the $\bigcup$ symbol lets on.

It's no surprise, then, that Branan and Erlewine (forthcoming) and Preminger (2019) do not embrace BFS as a minimal search algorithm, with Preminger defining a version of DFS and Branan and Erlewine making no firm decision between the two options. This is not to say that the authors are not aware of the problems of DFS that I outline above. On the contrary, Branan and Erlewine explicitly addresses these issues and they 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 algorithim—at least, not given the theretical assumptions of the current paper. Preminger proposes that specifiers are not searched, while Branan and Erlewine suggest that left-branches might 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 oppposite.

Ke (2019, pp. 47–49), on the other hand, attempts to split the difference by modelling BFS as 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, so it is difficult to assess its empirical and theoretical consequences. If such an algorithm can be shown to be adequate in those respects, it has the potential to be a better model of BFS—one without the need for auxiliary notions like tiers.

It would be a mistake, though, to declare DFS fully discredited on the basis these arguments. The theoretical and aesthetic appeal that I describe above must be answered and the hypotheses that Branan and Erlewine and Preminger put forth to rescue it have empirical backing. What is needed, though, is a principled theory that predicts rather than declares, for example, that the internal structure of a specifier is inaccessible to minimal search. Since I know of no such theory, I will assume that DFS is inadequate as a model of minimal search.

3.2 A formal definition of Agree

If and when an instance of Probe retrieves a goal, that goal must be modified—at least according to most versions of Agree.[5] More precisely, the Goal must be modified in place. That is, if goal G is in position Q in stage S_i, then the modified goal G' must be in position Q in stage S_{i+1}. Furthermore, if copies of G are in multiple positions $(Q, Q', Q''\ldots)$ in S_i, then copies of G' must be in 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. Thus we can define Agree as in (51).

(51) For lexical item P, syntactic object SO=$\{P, \ldots\}$, and feature type F, and lexical-item G, the sole member of Probe(SO,P,F),

$$\text{Agree}(\text{SO, P, F}_\text{v}) = \begin{cases} \text{Value(SO,}\langle\text{F, v}\rangle\text{)if SO=G} \\ \text{SO if SO is a lexical item token} \\ \text{Merge(Agree(A, P, F}_\text{v}\text{), Agree(B, P, F}_\text{v}\text{)) for A,B} \in \text{SO such that A} \neq \text{B} \end{cases}$$

As defined, Agree is a non-minimal DFS—it has no notion of tiers, only differentiating lexical item tokens from complex syntactic objects. While minimalist considerations might suggest that a single search algorithm be selected for the grammar, DFS is ill-suited for Probe, as discussed above, and BFS is ill-suited for Agree. The reason we cannot use BFS for Agree is because Agree must retain the structure of its inputs—it needs to put things back where it found them—something that BFS cannot do. Consider, for instance, Tier 3 in (46)—a

[5] 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?(P, G, F) iff Probe(P, F) = G.

4-member set which could be reconstructed into a proper syntactic object a number of ways. Thus, we need both DFS and BFS to be active in the grammar.

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 (52) which immediately contain lexical item tokens bearing a valued feature $\mathrm{F_v}$ and which contain a lexical item token bearing an unvalued feature $\mathrm{F_0}$.

(52) $\{\mathrm{P_{F:v}}, \{\ldots \mathrm{G_{F:0}}\}\}$

In an Upward Valuation, the relevant features of P and G are swapped, as in (53).

(53) $\{\mathrm{P_{F:0}}, \{\ldots \mathrm{G_{F:v}}\}\}$

In order to capture Upward Valuation, then we need first modify the Match criterion of Probe as in (54), moving P to the second argument position.

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

$\mathrm{Probe_{UV}}$(SO, P, F) = Search(SO, (λx) (Match(x, P, F))).

Thus, $\mathrm{Probe_{UV}}$ gives a definite result {G} only if P contains an unvalued F feature and G contains a valued F feature. Since, by definition, the relevant unvalued feature in $\mathrm{Agree_{UV}}$ 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 $\mathrm{Agree_{UV}}$ in (55) look similar to that in (51).

(55) For lexical item P, syntactic object SO={P, . . . }, and feature type F, and lexical-item G, the sole member of $\mathrm{Probe_{UV}}$(P,F) and v the value of the F feature on G,

$$\text{Agree}_{\text{UV}}(\text{SO}, \text{P}, \text{F}_{\text{v}}) = \begin{cases} \text{Value(SO,}\langle\text{F, v}\rangle\text{)if SO=P} \\ \text{SO if SO is a lexical item token} \\ \text{Merge(Agree}_{\text{UV}}\text{(A, P, F}_{\text{v}}\text{), Agree}_{\text{UV}}\text{(B, P, F}_{\text{v}}\text{)) for A,B} \in \text{SO such that A} \neq \end{cases}$$

3.4 Feature Checking

Versions of Agree that causes feature checking rather than valuation assume that all formal features—*i.e.*, members of SYN-F—are valued, but must be checked by Agree. 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 (56) and (57), with semantic features retaining their definition in (24).

(56) 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*.

(57) 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 (58)

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

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

Finally, Check is a simple matter of flipping a 0 to a 1 as in (59).

(59) For a formal feature F_{v}=⟨c?, F, v⟩,

$\text{Check}(\text{F}_{\text{v}}) = \langle 1, \text{F}, \text{v}\rangle$.

These newly defined functions can be slotted into our formalized definitions of Agree, perhaps with a few other alterations, which I leave as an exercise for the interested reader.

3.5 Local Agree

Early minimalist theories of agreement (*e.g* Chomsky 1993) continued the GB assumption that agreement was limited to a spec-head relation. So, for example, subject-predicate agreement was assumed to occur because 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 ans specifically limited to the very top of object. The canonical case of spec-head agreement is the finite subject merged with the finite predicate, shown in (60)

(60) $\mathrm{TP} = \{\{\mathrm{D}, \ldots\}, \{\mathrm{T}, \ldots\}\}$

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 $\mathrm{Probe}_{\mathrm{Local}}$, then, as in (61).

(61) For feature type F, lexical item tokens P and G, and syntactic object SO={X, Y},

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

Since spec-head structures, especially those associated with Case and agreement, are often formed by Internal Merge, our final version of $\mathrm{Agree}_{\mathrm{Local}}$, much like long-distance Agree, will need to replace every instance of the object being valued/checked. Therefore, our final version of $\mathrm{Agree}_{\mathrm{Local}}$, like our baseline Agree in (51), will be recursively defined—the main difference between the two will be their respective Probe prerequisites.

Other changes must be made to Agree though. Recall, for instance, that, in order to account for ambiguous searches, Search was defined in (49) such that its output was a set of lexical item tokens, and Agree was defined in (51) so that it only proceeds when the output of Probe—a species of Search—is a singleton set. $\text{Probe}_{\text{Local}}$ does not have to account for ambiguous searches—either the appropriate G is the head of the specifier of P, or it isn't. Therefore, the Probe prerequisite of $\text{Agree}_{\text{Local}}$ must be rewritten. This is a relatively minor rewrite, but a rewrite nonetheless.

3.6 Summary

I 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 UG_{Agree}

With the Agree operation properly formalized, we can give a definition of UG_{Agree} in (62) and derivation in (63).

(62) Universal Grammar is a 7-tuple: $\langle$PHON-F, SYN-F, SEM-F, Select, Merge, Transfer, Agree$\rangle$

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

i. For all LI and k such that $\langle$LI,k$\rangle \in \text{LA}_1$, 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∈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, P, G and F:

a. $SO \in W_i$

b. SO immediately contains P

c. Probe(SO,P,F) = {G}

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

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 there full definition in intension. Agree is ultimately defined recursively, as is its prerequisite Probe, so such an expansion is not possible. This is a crucial difference between Agree and the other generative operations. While we could conceivably rank Select, Internal-, and External-Merge by complexity, such a ranking would be one of degree. Agree, however, with its recursive definition is a different kind of operation. 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, Agree entails Merge. This is reflected in definitions (51) and (55) 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.

Agree being dependent on Merge also raises a biolinguistic critique. Chomsky (2020, and elsewhere) proposes the following evolutionary narrative of the language faculty. The faculty of language (*i.e.*, Merge) evolved quite suddenly 40 000–50 000 years ago in humans as a purely internal instrument of thought. It was only later, after humans began migrating out of Africa, that externalized language emerged (Huybregts 2017). This narrative explains the fact that much, perhaps most, of our use of language is strictly internal to our individual minds—that language is independent of externalization. Or, put another way, this story of the evolution of the language faculty correctly predicts that the set of externalized linguistic objects (LOs)—*i.e.*, the set of expressions which have been actually spoken, signed, or written by actual humans—is a subset of the set of actually generated linguistic objects as in figure 1. The fact that Agree entails Merge suggests either that it emerged as part

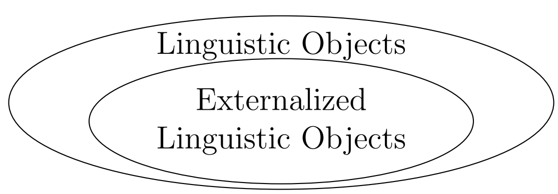


Figure 1: The relation between LOs and externalized LOs

of externalization—which I address later—or it emerged separately from both Merge and Externalization. The latter option includes two suboptions—either Agree emerged as an augmentation to Merge and Externalization emerged as an augmentation to Merge+Agree, or Agree and Externalization emerged as separate augmentations to Merge. The former option would predict that the set of Agreeing LOs is a subset of the set of LOs and a superset of the set of externalized LOs, as shown in figure 2. The latter option would predict that

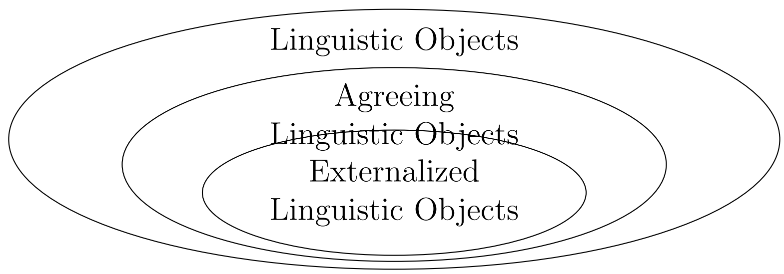


Figure 2: A possible relation between LOs, Agreeing LOs and externalized LOs

the set of Agreeing LOs and the set of Externalized LOs are each a subset of the set of LOs, though neither is a subset of the other, as shown in figure 3. Note that the overlap between

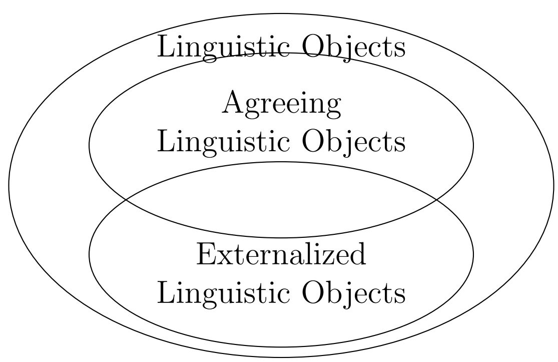


Figure 3: A possible relation between LOs, Agreeing LOs and externalized LOs

Agreeing LOs and Externalized LOs is not theoretically or logically guaranteed, but rather is an empirical fact. Each of these options predicts that non-external LOs can be divided into Agreeing and non-Agreeing LOs, while the latter further predicts that external LOs show the same division. These are, in principle, empirical predictions albeit not yet practically so, as it is not clear what non-Agreeing LOs, either internal or externalized, look like in this context.

4.1 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 computational operation must be closed under that operation. 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 it's 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 the case.

Consider the lexical item token X_k, defined in (64), which has only one syntactic feature, [F:0].

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

What about the result of applying Value to X_k, given in (65)?

(65) $\text{Value}(X_k, \langle \text{F}, \text{v}\rangle) = \langle\langle \text{PHON}_X, \{\langle \text{F}, \text{v}\rangle\}, \text{SEM}_X\rangle, k\rangle$
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 $\langle \text{F}, \text{v}\rangle \in \text{SYN-F}$. That is, the set of lexical item tokens is closed under Value only if the universal set of syntactic features in UG_{Agree} contains both valued and unvalued features.

However, if we hypothesize that SYN-F contains valued and unvalued features, we are faced with something of a theoretical quandary. In this system, language acquisition is 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. If this is the case, then there are 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 case, consider the English third person singular present agreement morpheme *-s*. Taking for granted that an English acquirer can give a proper phonological and semantic analysis of the morpheme, there are two possible lexical items they could construct, given in (66) and (67).

(66) $\langle$[z], $\{\langle\pi,3\rangle,\langle\#,1\rangle\}$, $\{\langle\mathrm{T},1\rangle\}\rangle$

(67) $\langle$[z], $\{\langle\pi,0\rangle,\langle\#,0\rangle\}$, $\{\langle\mathrm{T},1\rangle\}\rangle$

The lexical item in (66), would be the result of a surface analysis of the data, while the one in (67) would require a deeper analysis. So, in order to predict the acquisition of (67), we would need a theory of acquisition that systematically does not match lexical items to surface phenomena.

Supposing on the contrary, that we bite the bullet and allow for valued lexical items to be acquired, even if we stipulate that unvalued lexical items are also acquired, economy considerations would suggest that those unvalued lexical items would never be used. In such a situation, every complex expression of a language would be derivable in at least two ways—one that begins with a lexical array containing only unvalued lexical item tokens and one that begins with a lexical array containing only valued lexical item tokens.[6] Each derivation will have the same number of Merge steps and Select steps but the fist derivation will also have Agree steps, while the second will have no Agree steps. Thus, for any expression of a language, the second type of derivation will always have fewer steps than the first. So

[6] Setting aside the possibility of lexical items without syntactic features.

	Closed under Merge	Closed under Select	Closed under Agree
Syntactic Objects	Yes	Yes	No
Valued Syntactic Objects	Yes	Yes	No
Syntactic Objects $\cup$ Valued Syntactic Objects	Yes	Yes	Yes

Table 1: The closure properties of Merge, Select, and Agree

paradoxically, expanding our universal feature sets to allow for Agree in this way, effectively rules out Agree.

To get out of this paradox, we could simply expand the domain of Merge, Select, and Agree to encompass the union of the set of lexical items and the set of valued lexical items. This would fix the problem in an engineering sense—we would be able to derive expressions in our formalism—but it would only serve to formalize the theoretical concerns that I have been addressing. It would do so because it highlights the fact that UG with only Merge and Select is a fully self-consistent system whose domain must be augmented to accommodate Agree. This situation, which can be seen in table 1, is hardly surprising considering the very nature of the operations—Merge combines objects without changing them, Select rearranges objects without changing them, Agree changes objects.

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, theorist 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.[7] Since C&S's

[7] Wurmbrand (2014) contains the most explicit argument in favour of the latter stance, but see Boeckx (2010) for a broader discussion of the schism.

formalization and my extension of it assume that all operations, except perhaps Transfer, are free, I will not discuss the former way out of the dilemma. Rather, 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 (63), the derivation is a computational procedure and, therefore, is strictly incremental. That is, the validity of a given stage S_n (n≠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 a stage S_n (n≠1) depends on its two preceding stages S_{n-1} and S_{n-2}. 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 a line in our derive-by-Merge clause in (68).

(68) 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,

c. $\langle W_i, LA_i \rangle$ 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, P, G and F:

a. $SO \in W_i$

b. SO immediately contains P

c. $\text{Probe}(SO,P,F) = \{G\}$

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

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 (69) with an empty workspace and a lexical array containing at least X and Y.

(69)
$$\begin{aligned} S_1 &= \langle W_1, LA_1 \rangle \\ &= \langle \{\}, \{X, Y, Z \ldots\} \rangle \end{aligned}$$

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

(70)
$$\begin{aligned} S_2 &= \text{Select}(X, S_1) \\ &= \langle \{X\}, \{Y, Z \ldots\} \rangle \end{aligned}$$

(71)
$$\begin{aligned} S_3 &= \text{Select}(Y, S_2) \\ &= \langle \{X, Y\}, \{Z \ldots\} \rangle \end{aligned}$$

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 (68) 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 (68), 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 (68) holds.

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 to 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 that, though it has fallen out of fashion, faces fewer theoretical hurdles than long-distance Agree in my opinion.[8]

4.3 Computational Complexity

With our definitions of the derivation in (20) and (63) we can give a quantitative estimate of the computational complexity of a given derivation, and with that, a measure of the complexity of the grammars overall. As is common in computer science, we will use time-complexity as a proxy. The time complexity of an algorithm is a measurement of how the run-time an algorithm—the length of time it takes to run the algorithm—increases relative to the size of its input.

To assess time complexity we must first identify the primitive operation(s) of an algorithm, which we assign a runtime of 1, and the primitive unit of data, which we assign a size of 1. In our derivations the primitive operations are Merge and Select as neither is defined in terms of the other, while Agree is defined in terms of Merge. Each instance of Merge or Select, then, will incur a time cost of 1—the time cost of Agree will be calculated below, and that of Transfer will be ignored. The input size will be a measure of the size of the derived syntactic object which will have two components—the number of lexical item tokens L, and

[8]Strictly speaking, covert A-movement has fallen into disuse. Covert $\bar{A}$-movement operations, like Quantifier raising, and covert *Wh*-movement in *wh-in-situ* languages, are still considered respectable hypotheses.

the number of syntactic objects J. The two numbers are related only insofar as they limit each other (L≤J). In practice, though, we will care less about J than the number of derived syntactic objects M=J-L. So, the objects in (72) all have different L,J, and M values.

(72) a. A (L=1,J=1,M=0)

b. {A, B} (L=2,J=3,M=1)

c. {A, {A, B}} (L=2,J=4,M=2)

d. {B, {B, {A, B}}} (L=2,J=5,M=3)

e. {C, {B, {A, B}}} (L=3,J=6,M=3)

Before we assess $\mathrm{UG}_{\mathrm{Agree}}$, though, we will consider plain UG to see how we would calculate the run-time of a given derivation. So, for a derivation D, the run-time R will be the sum of μ—the number Merge operations performed in D—and σ—the number of Select operations performed in D.

(73) $\mathrm{R}_{\mathrm{D}} = \mu + \sigma$ for UG

In order to calculate μ and σ, we step through each stage S_n of D, keeping a running tally of each operation.

(74) $$\mu_{\mathrm{S}_n} = \begin{cases} 0 & \text{if n=1} \\ \mu_{n-1} + 1 & \text{if } \mathrm{S}_n \text{is derived by Merge} \\ \mu_{n-1} & \text{otherwise} \end{cases}$$

(75) $$\sigma_{\mathrm{S}_n} = \begin{cases} 0 & \text{if n=1} \\ \sigma_{n-1} + 1 & \text{if } \mathrm{S}_n \text{is derived by Select} \\ \sigma_{n-1} & \text{otherwise} \end{cases}$$

Since each Select operation in a derivation is associated with a distinct lexical item token, σ for that derivation will equal L for the derived object. Similarly, each Merge operation in a derivation creates a distinct new syntactic object, so the μ for for that derivation will equal

M for the derived object. Therefore, under UG, the runtime of the derivation for a syntactic object will be J for that object. So, if we take J to be our measure of the input size for a derivation, we can see that UG derivations run in what is called linear time.

In order to assess $\mathrm{UG}_{\mathrm{Agree}}$ we need a way to measure the run-time of Agree. For simplicity's sake, I will not consider the run-times of Value, Match, or Probe, or rather, I will take them to be zero. So, this simplified Agree, when applied to a lexical item token, returns that token, and when applied to a derived object, recursively performs Agree on the members of the object and Merges the results. When applied to an object X then, Agree runs a Merge operation for each derived syntactic object in X.

We can define our running tally for Agree in (76) with the final calculation of run-time in (77)

(76) $$\alpha_{\mathrm{S}_n} = \begin{cases} 0 & \text{if n=1} \\ \alpha_{n-1} + \mu_{n-1} & \text{if } \mathrm{S}_n \text{is derived by Merge} \\ \alpha_{n-1} & \text{otherwise} \end{cases}$$

(77)

(78) $\mathrm{R_D} = \mu + \sigma + \alpha$ for $\mathrm{UG}_{\mathrm{Agree}}$

Since $\mathrm{UG}_{\mathrm{Agree}}$ does not specify when Agree applies, it allows for derivations where Agree does not apply at all. These cases will run in linear time, and will be our lower bound for time complexity. As our upper-bound, consider the cases in which every instance of Merge is followed immediately by an instance of Agree. Since μ determines the rate of increase for α and μ increases linearly during the course of the, α will increase quadratically, and therefore, R will increase quadratically relative to the number of stages. The run-time of such a derivation is demonstrated in figure 4. Since the number of stages here is proportional to the size of the derived object, the time-complexity of this type of derivation is also quadratic.[9]

[9]More precisely, the run-time of this type of derivation as a function of object size is resembles the triangular number series (1).

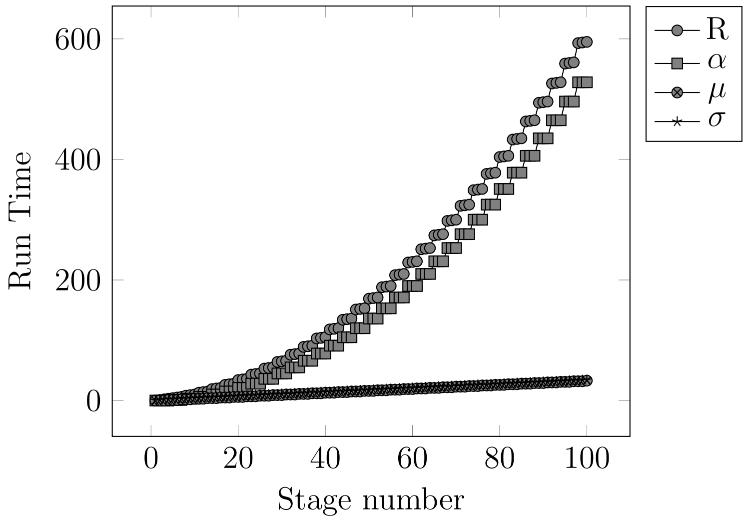


Figure 4: The run-time of a derivation in $\mathrm{UG}_{\mathrm{Agree}}$ following a Select-Merge-Agree cycle

Of course, the Select-Merge-Agree cycle that this assumes is not a realistic characterization of an actual syntactic derivation for a number of reasons. For one, it represents a derivation with only External Merge, while the overwhelming evidence suggests that actual expressions are always derived with a mix of internal and external. Also, it is likely not the case that every instance of Merge is followed by an instance of Agree. For example, cyclic movement through non-licensing positions could be argued to involve Merge but not Agree. Even including all of these caveats, the facts that the run-time of a single instance of Agree is proportional to the size of the object it operates on and that the size of that object steadily increases throughout any derivation mean that no derivation which includes more than one non-consecutive instance of Agree will operate in linear-time.

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

(1) $\sum_{i=0}^{n} \frac{i(i+1)}{2}$

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 (79).

(79) 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 increased computational complexity of UG_{Agree} discussed above, then, is predicted by Chomsky's conjecture that the NTC is linked to computational efficiency. 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 (80)

(80) A stage is a triple $S = \langle LA, W, L\rangle$, 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 $\langle LI_k, LI'_k\rangle$, 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 (51) does. Therefore it does not need to be recursively defined, and it does not need to refer to Merge in its definition. As a result, it does not carry the same time-costs as standard Agree.

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 P and G are lexical item tokens and, for some feature F, Match(P,G,F)=1. Further suppose that stage S_n in derivation D is derived by Merge(P, X), where X contains G and no lexical item token H, such that Match(P,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 {P, X} 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 P to Agree with G for F.[10] 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 don't need Agree as a derivational operation.

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

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

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.

(81) 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.[11] The form of the argument is given in (82)

(82) 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.

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

[11] 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 had. See Hornstein (2009) for a proposal that predicts the effects of tucking-in without tucking-in.

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."[12]

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 (82), which Preminger claims as an empirical result, that canonical subjects in non-quirky-subject languages should always trigger 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 (83), where the underlined subexpression is could be or has been considered a subject in English.

(83) 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.

f. We believed them to be a capable team.

[12]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).

I believe that it is quite safe to label *the city* in (83a) as a canonical subject[13]—it is the specifier of TP and it triggers φ-agreement on the finite auxiliary. On the other hand, the existential associate *unicorns* in (83b) is likely not in a canonical subject position.[14] In fact, existential associates not being in canonical subject position gives force to the second premise of (82)—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.[15]

This leaves us with non-finite subjects in (83c) to (83f). 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 (83c) as a canonical subject, since it is not a specifier to a TP, leaving us with the null subjects in (83d) to (83e) and the ECM subject in (83f). 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,[16] the Uniformity Principle—Preminger's third premise—demands that we treat movement to the specifier of finite T as a grammatical process, which, in the current system, means treating it as a derivational procedure distinct from Merge, Select, Agree and Transfer. So, if we keep strictly to the theory assumed in this paper, Preminger's

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

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

[15]The expletive *there* in (83b) seems to be in canonical subject position—if *unicorns* was there it would be the canonical subject—but it does not trigger φ-agreement. This, however, does not contradict (82b), which links φ-agreement with movement to canonical subject position, not to the position itself, if we assume that expletives are inserted in subject position, not moved there.

[16]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.

argument does not go through.[17]

To recap, Preminger's argument as given in (82), while 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[18]—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.

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 ans SEM features of LIs. The choice to formalize features as type-value pairs, though, does have relevant consequences.

[17]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.

[18]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,

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 (84).

(84)

$$\begin{array}{c} \pi \\ | \\ \textsc{participant} \\ | \\ \textsc{addressee} \end{array}$$

One formal definition of feature that would capture this is given in (85), with 2nd person feature formalized as in (86)

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

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

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 a probe can Match a goal that is less specified than it or if goals should be more specified than probes—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 (49) nor our final definition of Agree in (51)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, had 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 can, in a way, be measured and given an objective evaluation—in section 4, I showed that derivations with Agree were in a different complexity class than those without Agree, and that Agree is incompatible with the NTC, a central minimalist tenet. I further showed that, while the set of syntactic objects, as defined by Collins and Stabler (2016), is closed under Merge, it is not closed under Agree without making some ad-hoc modifications to our theory.

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, Susana (2003). 'Phi-syntax: A theory of agreement'. Doctoral dissertation. University of Toronto.

Béjar, Susana and Milan Rezac (2009). 'Cyclic agree'. In: *Linguistic Inquiry* 40.1, pp. 35–73.

Bjorkman, Bronwyn and Hedde Zeijlstra (2014). *Upward Agree is superior.*

Boeckx, Cedric (2010). 'Reflections on the plausibility of crash-proof syntax, and its free-merge alternative'. In: *Exploring Crash-Proof Grammars*. Ed. by Michael T. Putnam. Vol. 3. Language Faculty and Beyond. John Benjamins Publishing Company, pp. 105–124.

Branan, Kenyon and Michael Yoshitaka Erlewine (forthcoming). 'Locality and (minimal) search'. URL: `https://ling.auf.net/lingbuzz/005791`.

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

Chomsky, Noam (1965). *Aspects of the theory of syntax*. Cambridge: MIT Press.

— (1993). 'A minimalist program for linguistic theory'. In: *The view from Building 20: Essays in linguistics in honor of Sylvain Bromberger*. Ed. by Ken Hale and Samuel Jay Keyser. MIT press.

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

— (2004). 'Beyond Explanatory Adequacy'. In: *Structures and Beyond*. Ed. by Adriana Belletti. The Cartography of Syntactic Structures 3. Oxford University Press, pp. 104–131.

— (2007). 'Approaching UG from below'. In: *Interfaces + recursion = language? Chomsky's minimalism and the view from syntax-semantics*. Ed. by Uli Sauerland and Hans-Martin Gärtner. Mouton de Gruyter Berlin, pp. 1–29.

Chomsky, Noam (2013). 'Problems of projection'. In: *Lingua* 130, pp. 33–49.

— (2020). 'The UCLA Lectures'. URL: https://ling.auf.net/lingbuzz/005485.

Collins, Christopher and Erich Groat (2018). 'Copies and Repetitions'. URL: https://ling.auf.net/lingbuzz/003809.

Collins, Christopher and Edward Stabler (2016). 'A Formalization of Minimalist Syntax'. In: *Syntax* 19.1, pp. 43–78. DOI: 10.1111/synt.12117. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/synt.12117. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/synt.12117.

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

Harbour, Daniel (2007). *Morphosemantic number: From Kiowa Noun Classes to UG Number Features*. The Netherlands: Springer.

Harley, Heidi and Elizabeth Ritter (2002). 'Person and number in pronouns: A feature-geometric analysis'. In: *Language* 78.3, pp. 482–526.

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

Huybregts, M.A.C. (Riny) (2017). 'Phonemic clicks and the mapping asymmetry: How language emerged and speech developed'. In: *Neuroscience & Biobehavioral Reviews* 81. The Biology of Language, pp. 279–294. ISSN: 0149-7634. DOI: https://doi.org/10.1016/j.neubiorev.2017.01.041. URL: https://www.sciencedirect.com/science/article/pii/S0149763416305450.

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

McCawley, James D. (1968). 'The Role of Semantics in a Grammar'. In: *Universals in Linguistic Theory*. Ed. by E. Bach and R. Harms. New York, NY: Holt, Rinehart & Winston, pp. 124–169.

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

Preminger, Omer (2018). 'Back to the Future: Non-generation, filtration, and the heartbreak of interface-driven minimalism'. In: *Syntactic Structures after 60 Years: The Impact of the Chomskyan Revolution in Linguistics*. Studies in Generative Grammar [SGG]. De Gruyter, pp. 355–380.

— (2019). 'What the PCC tells us about "abstract" agreement, head movement, and locality'. In: *Glossa: A Journal of General Linguistics* 4.1, p. 13. DOI: 10.5334/gjgl.315.

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

Starke, Michal (2010). 'Nanosyntax: A short primer to a new approach to language'. In: *Nordlyd* 36.1, pp. 1–6.

Wurmbrand, Susi (2014). 'The merge condition : A syntactic approach to selection'. In: *Minimalism and Beyond: Radicalizing the interfaces*. Ed. by P. Kosta et al. Language Faculty and Beyond. John Benjamins Publishing Company, pp. 130–166.

Zeijlstra, Hedde (2012). 'There is only one way to agree'. In: *The linguistic review* 29.3, pp. 491–539.