Minimal Search in Agree and Labeling

Alan Hezao Ke

Michigan State University

Department of Linguistics, Languages and Cultures

kehezao@msu.edu

Abstract

This paper develops a formal definition of Minimal Search to evaluate the idea that Agree and labeling could be reduced to Minimal Search. Different aspects of the search algorithm in Minimal Search, i.e., breadth-first vs. depth-first search, parallel vs. serial search, global vs. modular search are compared, and reasons for choosing between each of these pairs are given based on detailed examinations of their theoretical and empirical consequences. This paper argues, based on the formal definition of Minimal Search, that Agree and labeling can only be partially unified by Minimal Search: the search algorithms in Agree and labeling can be unified by Minimal Search, but the values of the search targets and search domains are determined by Agree and labeling independently. This paper (re)defines Agree and labeling based on Minimal Search to capture both the similarities and differences between these two operations.

Keywords: Minimal Search, Agree, labeling theory, 3rd factor, search algorithm

I am deeply indebted to Samuel Epstein and Richard Lewis for their insightful comments and their generous support during the development of the core ideas presented in this paper. I should also thank Acrisio Pires, Andrew McInnerney, Jonathan Brennan, Yushi Sugimoto, Hisatsugu Kitahara, and Daniel Seely for their discussions and helpful feedback. All remaining errors are mine. This article is in memory of Samuel Epstein.

1. Introduction

An influential idea in recent Minimalist theory is that Agree and labeling are simply Minimal Search. Minimal Search has been proposed to be a 3rd factor principle (Chomsky 2005), as it is a principle of efficient computation which language, a computational system, may very likely observe.

This assumption has an important conceptual motivation. Currently, there is arguably an emerging consensus that the syntactic component of the language model includes three operations: Merge, Agree and labeling. If Agree and labeling can be removed from the system and be replaced by the 3rd factor Minimal Search without a significant empirical loss, the system would be considerably simplified.

Chomsky (2013) has made this assumption explicit. Chomsky (2013) writes, "The simplest assumption is that LA (labeling algorithm) is just Minimal Search, presumably appropriating a third-factor principle, as in Agree and other operations." (p. 43). In Chomsky (2015a), he continues this discussion, "Optimally, projection should be reducible to a labeling algorithm LA, a special case of Minimal Search (like Agree), which in turn falls under MC (minimal computation)." Chomsky therefore hypothesizes that an optimal system would have Agree and the labeling algorithm derived from a more general, independently motivated third-factor principle, i.e., minimal computation. This conjecture, if successful, could unify the labeling algorithm and Agree by resorting to an arguably third-factor operation, Minimal Search. Consequently, since Minimal Search as a third factor is freely and universally available, it might be the case that labeling algorithm and Agree are also freely available (Chomsky 2005).

A serious implementation of the above conjecture is best conducted with a formal definition of Minimal Search. However, to the best of my knowledge, Minimal Search has not been formally defined to cover both labeling and Agree. The concept "Minimal Search" is an interaction of two important concepts: minimality (of computation) and a search algorithm. The idea of minimality in Agree (as we will see in Chomsky's (2000) definition of Agree and relevant discussion below) and labeling has gained significant attention in recent studies (Chomsky 2013; Epstein et al. 2014; Chomsky 2015a; Larson 2015; Chomsky 2019b; Ke 2019; Aycock 2020; Branan and Erlewine to appear; Epstein et al. 2020, among others). On the other hand, the discussion of search algorithm in Minimal Search in the literature, although interesting and insightful, is usually not explicitly based upon a formally defined algorithm (Krivochen 2021), with exceptions such as Ke (2019), EKS (Epstein, Kitahara and Seely) (2020), and Aycock (2020). In addition, Minimal Search is mostly considered for either Agree or labeling, not for both (e.g., EKS (2020) defines Minimal Search only for labeling). In this paper, a formal definition of Minimal Search for both Agree and labeling, based on which I will argue that labeling and Agree cannot be completely reduced to Minimal Search; they can only be partially unified by Minimal Search. On the one hand, the search algorithm in labeling and Agree can be captured by the current definition of Minimal Search. On the other hand, labeling and Agree are essentially different in their search target and search domain; they are thus needed for independent reasons and they serve distinct purposes. Various aspects of the search algorithm in Minimal Search, i.e., breadth-first vs. depth-first search, parallel vs. serial search, global vs. modular search are compared, and reasons for choosing between each of these pairs will be discussed in detail. I will then briefly discuss the implications of this particular definition with respect to Agree, labeling, and other empirical domains.

2. Defining Minimal Search

2.1 Insights from previous studies

As previously mentioned, Chomsky (2013) does not give a formal definition of Minimal Search; however, he does illustrate how Minimal Search works in labeling and Agree by examples. Let us consider some examples for labeling as in (1a, b).

(1) a. $\{_\alpha$ kick, {the ball}}

b. $\{_\beta$ $\{_\delta$ a, {little boy}}, $\{_\kappa$ is, {kicking the ball}}}

It seems that for the labeling of the set α in (1a), Minimal Search needs to look into α, and only the head member *kick* is returned. In this case, Minimal Search seems to look for a head and ignore *the ball*, the set member of α. This is reasonable because heads can be considered bundles of features, and sets do not carry features,[1] if we do not assume feature projection or percolation, following Chomsky (2013) and Chomsky (2015b). There is thus a natural distinction between heads and sets. In other words, Minimal Search in this case could search for any features. As long as it finds a feature, it returns the head bearing that feature, and then the search is immediately terminated. If Minimal Search is not terminated at this point and can continue to look into {the ball}, *the* would be returned as the label in addition to *kick*, which is not desirable. Therefore, this establishes a very important principle in Minimal Search, which is previously implicitly assumed but unnoted, as far as I know: Minimal Search terminates whenever a target is found. I will return to this principle later.

For (1b), Minimal Search needs to find a label for the set β, and thus looks into it. It cannot find any feature in β; instead, two sets are found. As a result, Minimal Search looks into the two sets, δ and κ. The heads it returns for these two sets are *a* and *is*. Note that this result is only possible if Minimal Search looks into δ and κ simultaneously. Otherwise, since Minimal Search terminates as soon as a target is returned (as pointed out in the previous paragraph), if Minimal Search serially searches into δ and then κ, only *a* will be returned. If, on the other hand, Minimal Search serially searches into κ and then δ, only *is* will be returned. Under serial search, there is no chance for *a* and *is* to be returned simultaneously. This is by no means a welcome result. Therefore, in order for Chomsky's labeling system to work as intended, we must adopt a search algorithm that can look into the two sets simultaneously (i.e., by applying parallel search, not serial search, as I will argue in more details later).

Furthermore, according to Chomsky (2013), the features in agreement between *a* and *is*, that is, the ϕ-feature pair <3SG, 3SG>, will be taken as the label of β. Again, the search needs to be terminated once the heads (*a* and *is*) are found and their prominent agreeing ϕ-features are extracted; otherwise, the head of *little boy* (whatever it is) and that of *kicking the ball* will also be returned as labels, which is not what we want.[2]

[1] I should thank Hisatsugu Kitahara for bringing this to my attention in one of our informal discussions.

[2] This strongly suggests that relevant ϕ-features must be all accessible on D; otherwise Minimal Search would need to search into the complement of D to find all the phi-features. If we assume DP is a phase and

Note that based on the labeling operation for structures such as (1b), Chomsky (2013, 2015b) could potentially derive the EPP (the Extended Projection Principle), which requires every (finite) TP to have a subject in its specifier position, potentially with additional assumptions.[3] This is because when the *v*P-internal subject is at its base-generated position, the *v*P cannot be properly labeled given that the *v*P-internal subject and the *v* head do not share agreement features, according to Chomsky (2013, 2015b). A solution to this problem is to move the *v*P-internal subject to a higher position. Since T agrees with the *v*P-internal subject, an agreement relation which will be discussed in the next paragraph, the *v*P-internal subject can move to the specifier of T′ (={T, *v*P}), where a legitimate label, i.e., $<\phi, \phi>$, is possible.

Now the question is how T can agree with the *v*P-internal subject and how Minimal Search is involved in this agreement. This is accomplished by the operation Agree. Below is Chomsky's (2000) definition of Agree, with minor modifications due to Chomsky (2004)[4]:

(2) *Definition of Agree* (Chomsky 2000:122)

Agree is a syntactic operation taking place between a probe P and a goal G in the domain of P, D(P), between which a Matching relation holds.

the phase head complement is transferred at the point when the phase head enters in the derivation (Chomsky 2000), *little boy* would have been transferred at the derivational step in (1b) and thus would be no longer accessible to Minimal Search. In this circumstance, we will still have to assume that all ϕ-features must be available on D, since D is the only accessible head in DP. By contrast, the standard theory of transfer (Chomsky 2008) does not remove the *v* head of *kicking the ball* and it should be available for search: this will cause unexpected problems to the labeling of (1b) if search is not terminated once a target is returned.

[3]See also Moro (1997) and Ott (2011). Chomsky (2015b) works out the derivation of the EPP for finite TPs, as he assumes that the T head in English is too weak to be the label of a TP, and therefore requires a SpecTP to be in an agreement relation with T, so that the features in agreement (the $<\phi, \phi>$ pair) can be the label. A non-finite T then does not require a SpecTP because they cannot be in an agreement relation.

[4]Chomsky (2000) considers matching as feature identity. However, Chomsky (2004) points out correctly that the matching between an uninterpretable feature and a corresponding interpretable feature is not feature identity but feature non-distinctness. I revised this part of definition as "matching is identity of feature attributes," excluding feature values in matching, to address the problem of the original definition of Agree.

a. Matching is identity of feature attributes;

b. D(P) is the sister of P;

c. Locality reduces to "closest c-command";

(3) *Definition of closest c-command*

A matching feature G is closest to P if there is no G′ in D(P) matching P s.t. G is in D(G′).

Three aspects of the above definition of Agree are worth noting: (i) Agree requires a probe, which will be the search target in my definition of Minimal Search below; (ii) the probe looks into a domain, which will be the search domain in my formalization; (iii) Agree observes a locality constraint, namely, closest c-command, which is likely what "minimal" means primarily in the term "Minimal Search."

To illustrate Chomsky's definition of Agree, let us use (4) as an example. In addition, a direct comparison between Agree in (4) and labeling in (1) may help us find out what exactly the mechanism is that is shared between labeling and Agree. (4a) is the derivation step before the T head, *be*, agrees with the D head of the *v*P-internal subject *this student* in (4b).[5]

(4) a. $[_{T'}$ be $[_{vP}$ $[_{DP}$ this student$]_j$ $[_{v'}$ writing the dissertation]]]

b. $[_{TP}$ $[_{DP}$ this student$]_j$ is $[_{vP}$ t_j $[_{v'}$ writing the dissertation]]]

After *be* merges into the derivation, as shown in (4a), a Minimal Search is initiated, and it looks into the *v*P set for a head bearing valued ϕ-features corresponding to those unvalued ones on *be*. No heads, but two sets, i.e., the DP and v', are found. So Minimal Search has to continue to search into these two sets. Both D and *v* are found. I assume *v* does not bear matching ϕ-features, either because ϕ-features in English are base-generated at V or feature inheritance from *v* to V has been completed at this derivation step (Richards 2007). Finally, the head *the* in the DP is returned, and Minimal Search terminates.

[5] I assume that all relevant ϕ-features are accessible on the D head, although not all ϕ-features are base-generated on D, following Sigurðsson (2017). See Footnote 2 for another reason why it is necessary for ϕ-features to be accessible on D from the perspective of the labeling algorithm.

Let us now summarize the roles Minimal Search plays in labeling and Agree. First, it looks into a set for some specific features (Agree) or any kind of features (labeling). Second, it applies iteratively to every available set, and it is terminated whenever a target is returned. To state it more formally, Minimal Search is a search algorithm that looks into a certain *search domain* (sets) for certain *search target* (features). The algorithm will find a syntactic object X before the syntactic objects c-commanded by X, as implied by Chomsky's definition of Agree in (2) which relies on the concept of closest c-command. The search is minimal because it is terminated as soon as a target is returned.[6]

2.2 The proposal

Since Minimal Search is considered a 3rd factor, we expect it to be found also in domains other than language. Below we will discuss relevant search algorithms in visual search and computer science, which may shed light on what search algorithm Minimal Search should be like in language. Let us consider first search algorithms employed in visual search. Suppose we are looking for a specific type of animal in box α. This is an instance of guided visual search where search is guided by features (Wolfe 2010), and this type of search might be serial or parallel or a mixture of these two. As shown in Figure 1, two boxes, β and γ, are contained inside α, and then δ is embedded in β and ρ in γ. Imagine that all the boxes have an opaque cover which must be removed before the searcher can see inside.

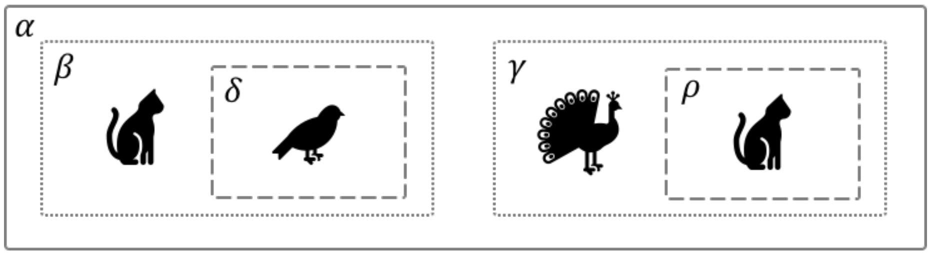


Figure 1: Visual Minimal Search

[6]Similar conceptualization of minimality has been previously formalized in Agree and Move, with a representational flavor, such as closest c-command, Minimal Link Condition (Chomsky 1995), and Relativized Minimality (Rizzi 1990). See Branan and Erlewine (to appear) for relevant discussion.

What is a search algorithm that is appropriate for visual search in this situation? Well, even before we look for an appropriate algorithm, we must first know what things/features we are searching for. That is, we must have a search target in mind. Suppose our search target is any cat. We then also need to determine the domain(s) we search into. The selection of search domain interacts with the search algorithm. If our search is serial, after searching into α, we may first search into β and then γ, or the other way around. However, if we can distribute our cognitive resources for different searches and thus be able to conduct searches in parallel, then an alternative algorithm is that we open and search into box β and γ simultaneously (cf. Pylyshyn and Storm 1988 for such a parallel visual search).[7]

There is a problem. In β you find a cat and in γ a peacock. Do you terminate search once you find the cat in β, or do you continue to search for a cat in ρ? This depends on the nature of your search. For serial search, there is probably no reason to continue the search once a target is returned. For parallel search, if the results of search are sent to your central processing systems, then you are aware that the search target has been found. You will terminate your search as you have achieved your search goal. By contrast, if each search in your parallel search (e.g., the search into β vs. the search into γ) is implemented in Fodorian modules that are informationally encapsulated and are of limited central access (Fodor 1983), then you are not aware of the results of searches that are carried out in parallel, and thus you will search into ρ further. Let us call the former search algorithm the global search algorithm, and the latter the modular search algorithm. The one I will adopt in the definition of Minimal search below is the global search algorithm. In other words, Minimal Search in Syntax terminates search whenever a higher syntactic object is returned, potentially consistent with what Chomsky states in Chomsky (2019a): "[D]on't use deep search if minimal search already works."

If the modular search algorithm is adopted, we would need to search into every box in Figure

[7]Compared to parallel search which generally finds the target faster, serial search in this instance may reduce the number of boxes that we need to open if we *happen* to start the search by looking into β and find a cat right away. However, this accidental nature of efficiency should not be in principle a basis of an argument for serial search. I will argue later that Syntax by nature may favor parallel search over serial search.

1 if there are more embedding boxes. That is, more boxes organized in a hierarchical structure will add to the search list and will lead to a massive search. Note that massive search is conducted even after the target has been found. This is intuitively not reasonable for visual search given Figure 1, and it is surely not computationally efficient in the sense that it involves much redundant search, incompatible with the idea that Minimal Search is a 3rd factor computational principle. Further linguistic evidence against modular search will be provided in Section 2.6.

Therefore, we want Minimal Search in the visual domain to be terminated as soon as the search target is returned, and this applies to both parallel and serial search. This is consistent with what we have seen in Section 2.1 regarding the search algorithm in labeling and Agree. In addition, the visual search example also confirms the necessity and importance of search target and search domain.

With what we have learned from Minimal Search in the visual domain, let us now turn to a definition of Minimal Search in Syntax, and we will turn back to the discussion of search algorithm from the perspective of computer science. As shown in (5), the current definition of Minimal Search includes a search algorithm that applies iteratively to a search domain (SD) to look for a search target (ST).

(5) *A formal definition of Minimal Search*

MS = <SA, SD, ST>, where MS = Minimal Search, SA = search algorithm, SD = search domain (the domain that SA operates on), ST = search target (the features that SA looks for).

Search Algorithm (SA):

a. Given SD and ST, matching against every head member of SD to find ST.

b. If ST is found, return the heads bearing ST and go to Step (c); Otherwise, get the set members of SD and store them as a list L.

   i. If L is empty, search fails and go to Step (c); otherwise

   ii. assign each of the sets in L as a new SD and go to Step (a) for all these new SDs in parallel.

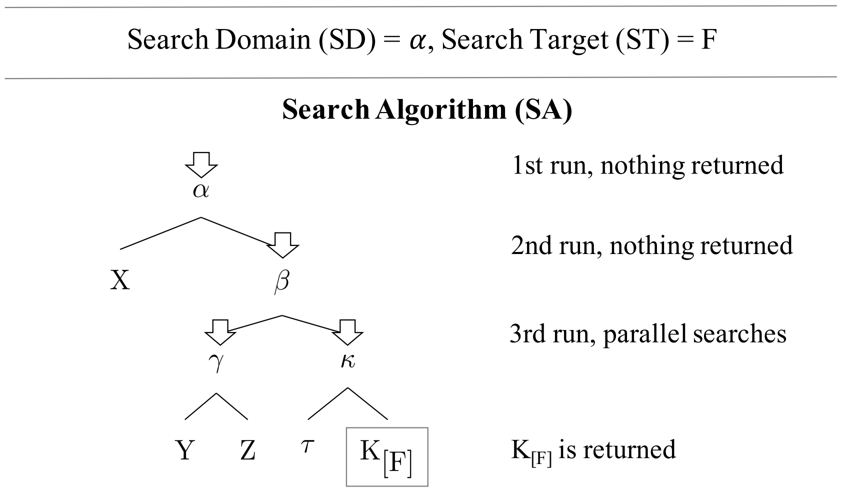
c. Terminate search.

The description of the search algorithm in (5) can be spelled out more explicitly, e.g., in terms of pseudo-codes, which I include in the appendix. Parallel instead of serial search is adopted in the definition, the reason of which will be discussed in detail in Section 2.4. Such an algorithm can be implemented in a programming language that allows parallel operations.

The definition uses lists instead of sets to store the new SDs and the returned heads. Lists are usually associated with linear order; however, I do not assume precedence relations between the SDs or the returned heads in a list.

2.3 An example

As an illustration of the Minimal Search as just defined, let us walk through an example. As shown below in Figure 2, a Minimal Search is initiated to search for an ST = feature [F], in the SD = set α.



Search Domain (SD) = α, Search Target (ST) = F
Search Algorithm (SA)
α
X
β
γ
κ
Y
Z
τ
$\boxed{K_{[F]}}$
1st run, nothing returned
2nd run, nothing returned
3rd run, parallel searches
$K_{[F]}$ is returned

Figure 2: An illustration of Minimal Search

Three runs of search are conducted before the target is found. In the first run, Minimal Search looks into α. X, the head member of α, is returned by the first run of search. However, since it does not bear the ST [F], and its co-member is a set, the search is not terminated. The set member of α, that is, the set β, is then assigned as a new SD.[8] This is the second run. The second run of search

[8]Note that the Minimal Search defined in (5) does not allow a single search applying to different sets. In order to search into a different set, a new Minimal Search must be initiated taking that set as a new SD.

finds only set members, i.e., γ and κ, and these two set members are stored as a list L. The two sets in L are then assigned as the new SDs for the third run of search, and two independent Minimal Searches are initiated in parallel. The head $K_{[F]}$ is found inside the set κ as it bears the ST. $K_{[F]}$ is finally returned by the search algorithm. The search over the set γ fails and is terminated without returning a result.

It is worth noting that the minimality in the (breadth-first) search is naturally (and partially) captured by *storing sets as a list L* in the definition (5b), not by, for instance, keeping track of/counting the levels of sets that Minimal Search looks into. It is also crucial to note that this search algorithm is not simply computing the path lengths between syntactic objects and comparing them. To compute the path between two syntactic objects, we have to first know the specific positions of these two objects. The relevant syntactic objects could be located in any positions in a syntactic representation, as long as a path can be formed. By contrast, the search algorithm here makes no such look-ahead assumptions about the specific position of the target.[9] It simply makes a search into the SD to find the ST. All the sets which are at the same level will be stored in the same list, and new Minimal Searches are conducted to look into all these sets simultaneously in parallel.[10]

2.4 Why breadth-first not depth-first?

There are two popular search algorithms in graph or tree search in computer science, breadth-first and depth-first search. In the case of traversing a search tree, a breadth-first algorithm explores all the daughters of a starting node before it searches any deeper. In Figure 3, breadth-first search visits the daughters of the starting node N1 before moving on to any of the daughter nodes of N2 or any node deeper. After all the daughters of Node 1 are visited, it then continues to traverse all the daughter nodes of Node 2 and Node 3. So on and so forth, until all nodes in the search tree are

[9]Ke (2019) uses a critical property of this search algorithm, i.e., the SD and ST are independently assigned, to derive upward agreement with downward search. This seems not possible with path length calculation.

[10]Natural language may enforce restrictions on the amount of items that are stored in a search list (see Footnote 15).

visited and the algorithm is terminated.

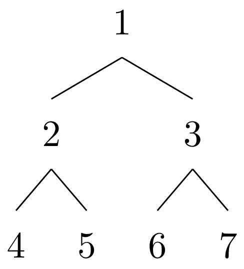


Figure 3: Breadth-first search

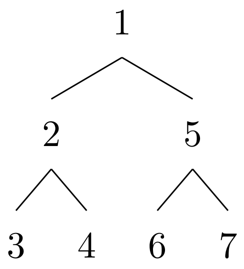


Figure 4: Depth-first search

On the contrary, a depth-first algorithm exhaustively searches down along certain nodes before backtracking. Take the tree in Figure 4 for an example. Starting from Node 1, depth-first search first visits a certain daughter of Node 1, e.g., Node 2, and then a daughter node of Node 2, e.g., Node 3, before it visits all the daughters of Node 2 and 3. Depth-first search reaches a terminal node (leaf node) there, so it backtracks to another daughter of Node 2, i.e., Node 4; then another daughter of Node 2, i.e., Node 5; and then a daughter of Node 5; so on and so forth, until it exhausts all the nodes dominated by Node 1.

Given an SD $\alpha = \{\beta, \gamma\}$, where Greek letters represent sets, a breadth-first search explores all the members of α before visiting the members of β and γ. On the other hand, a depth-first search goes to a member of α, for instance, β, and visits the members of β before exploring γ. The search algorithm in the definition of Minimal Search (5) is breadth-first (a parallel version).

Why do we adopt breadth-first search rather than depth-first search? Depending on the data structure, both the breadth-first and depth-first can be computationally efficient. The argument that this paper will provide for the breadth-first search is therefore not built on computational efficiency but on how the search algorithm can fit the language system. Indeed, a computational algorithm must fit the structure of the data it applies to.

Two considerations make a breadth-first algorithm a preferable option for labeling and Agree. First, the hierarchical nature of syntactic structures is one of the most important aspect of language and linguistic analysis, and the c-command relations are a most crucial instance of the hierarchical nature of language.[11] they capture Minimal Search must be compatible with this fundamental property of human language, and it must satisfy relevant restrictions. Breadth-first search respects the c-command relations as it looks into all nodes that are at a higher level before visiting any nodes at lower levels. By contrast, depth-first search does not exhaust all the items at a specific level first before visiting lower levels. The searching process in depth-first search cannot always track the hierarchical order of the levels, as there must be times that the algorithm traverse back from a lower level to a higher level. Indeed, depth-first search, to some extent, ignores some critical aspects of hierarchical structure. In particular, depth-first search have difficulties always capturing the asymmetric c-command relations (it primarily captures the containment relations): it may reach a c-commandee before accessing its c-commanders.[12]

For instance, in Figure 4, Node 3 and 4 are visited before their c-commander Node 5 is visited. On the other hand, if Node 5 is explored before Node 2, Node 6 and 7 will be visited before their c-commander Node 2 (and there is no principled reason to force the algorithm to visit Node 2 before Node 5 and vise versa). That is, no matter Node 2 or Node 5 is visited first, some c-command relations will be missed. It is thus not helpful to force the search algorithm to explore left-branches or right-branches first (cf. Milway 2021), either doing this in a systematic way or in a case-by-case manner, not mentioning imposing such a force might significantly complicate the search algorithm with language specific properties.

[11]C-command relations have been suggested to be derivable from Minimal Search (Ke 2019; Chomsky 2019b). However, c-command is still important here in that it is underlying accounts for a significant body of empirical generalizations (see Ke 2019 ch. 5 for a review). If a formalization of Minimal Search cannot capture the c-command relations in those attested cases, it misses the relevant empirical generalizations.

[12]C-command is defined as below:

(i) *Definition of c-command*

α c-commands β iff

a. neither α nor β dominates the other, and

b. the first branching node that dominates α also dominates β.

In principle, Minimal Search would need to return as early as possible a node containing the search target if the node is structurally closest to the starting node for search; structural closeness is defined in terms of c-command: If A c-commands both B and C, B is closer to A than C iff B c-commands C.[13] A breadth-first algorithm thus serves exactly this purpose. Due to the fact that depth-first search will visit a c-commandee before visiting its c-commanders, closest-c-command as found in Chomsky's definition of Agree in (2) cannot always be faithfully captured by Minimal Search if we adopt depth-first search.

The second consideration that favors the breadth-first search algorithm is that, importantly, depth-first search in fact would not give desired results with regard to labeling, if we assume that Minimal Search terminates at the point when the first target is returned. In the labeling of the {XP, YP} construction, the label will never be {X, Y} if depth-first search algorithm is adopted without further modifications, as it will be either that XP and all heads inside XP are visited before YP is explored, or that YP and all heads inside YP are explored before XP is visited. Neither of these two choices naturally give us {X, Y} or {ϕ, ϕ}. Making it worse, a depth-first Minimal Search may first visit XP before visiting H in the {H, XP} construction and returns X instead of H. The result is that the label for {H, XP} would be not H but X, which is theoretically and empirically inadequate.

2.5 Why parallel not serial?

We have argued that breath-first search rather than depth-first search is preferred for the Minimal Search algorithm. Why do we then employ a *parallel* breadth-first instead of a *serial* breadth-first search algorithm? The reason is that once set theory is adopted for structure building, linear order becomes a peripheral property. This is because the members of a set are standardly assumed to be symmetric with regard to their linear order. Linear order is not specified until syntactic objects are linearized at the SM interface. As a consequence, there is no principled reason to search into a specific member of a set before another set. One may argue that the algorithm can randomly search into any one of the set members. However, this will not give us the right result: in terms of {XP,

[13]Note that structural closeness can also be defined in terms of dominance (cf. Epstein et al. 2020). This is in fact captured by downward Minimal Search.

YP} constructions, if XP or YP is randomly searched into first, either X *or* Y, never both X *and* Y, will be returned, for I have argued at length that in general the algorithm must terminate once it encounters a target. Consequentially, we will never be able to find X and Y *simultaneously* in the case of {XP, YP} without adding further (likely unjustified) complications to the search algorithm.

Therefore, the non-ordering nature of set members favors the parallel breadth-first search algorithm, both theoretically and empirically, since such an algorithm looks into both set members simultaneously in parallel, without making a distinction between them; and a natural result of this parallel search is that X and Y in the {XP, YP} construction can be returned simultaneously.[14]

Finally, if the Syntax component can determining which member of a set is more "important" than the other, Minimal Search may make use of this information to guide the search. However, this requires the algorithm to "see" the features on the syntactic objects before it searches into them. The head-hood information, which probably gives some sense of "importance" to a certain member of a set, is not available until labels are computed at the time when syntactic objects are sent to CI for interpretation. Therefore, Minimal Search is not sufficiently informed to make such decisions before it actually carries out searches. [15]

[14]One might think of a particular implementation of the parallel search algorithm as a case of serial search. For instance, searching into {XP, YP} could potentially be implemented as searching into XP, and then YP, as long as the algorithm terminates only after the list containing XP and YP is exhausted. Note that this is only a convenience *implementation* of the parallel search algorithm. We may implement the algorithm in different ways: it could be in the way parallel sentence processing (MacDonald et al. 1994; Trueswell et al. 1993, 1994) or parallel visual search (see Footnote 15) is implemented in the brain. What the paper advocates is *not* about any technical implementations but about the nature of the search algorithm. We may conceptualize these as two distinct levels of inquiry based on Marr's (1982) three levels of analysis of complicated cognitive systems: what this paper proposes is at the computational and (primarily) algorithmic levels of analysis, whereas a particular implementation of Minimal Search is at the implementational level of analysis.

[15]The reader may think of computation burden as a drawback of the parallel breadth-first search algorithm, mainly due to the storage of multiple search domains in a list and relevantly the conduction of multiple searches in parallel (Aycock 2020). However, Minimal Search by definition is a third factor, and therefore it is not specific to language and it potentially does not take resources from the Syntax module but instead uses

2.6 Why global not modular?

In this section, I would like to address a potential question which touches on the parallel search aspect of the definition of Minimal Search in this paper: What if the parallel searches do not communicate?[16]

As mentioned in Section 2.2, a potential formalization different from the global search algorithm advocated in this paper is the modular search algorithm, according to which parallel searches do not communicate with one another regarding their search results. They are like modules whose information cannot be directly accessed from other modules. However, I have identified a main drawback of that approach, as illustrated with visual search: the algorithm does not terminate even after a search target has already been returned, due to the fact that under this algorithm searches do not communicate and they do not know after a target has been returned. Therefore, as we have seen in the example of visual search previously, the modular search algorithm will necessarily continue to exhaust all possible sets and subsets that are available for search even after a target has already be found, thus leading to tremendous unnecessary search.

Below I will highlight a few empirical problems associated with the modular algorithm in Minimal Search for Agree. The problems that are highlighted below are essentially caused by the very nature of modular search: it violates computational minimalism.

Imagine that T with [uϕ] searches for a head with matching features. Using modular search, the search target is set to ϕ-features attributes as those on T and the search domain is *v*P. The modular

general resources. Of course, general attentional resources are also restricted and limited, but Minimal Search can be carried out subconsciously without resorting to attention or even memory. Indeed, just like parallel visual search which is not dramatically slowed down by the increase of the amount of objects in the search array (Woodman et al. 2001; Horowitz and Wolfe 1998), Minimal Search may be not so strictly restricted by the amount of search domains and the amount of searches in parallel. However, I will also not go to the other extreme that Minimal Search is completely not restricted on the capacity of its search lists. In fact, language has natural mechanisms to reduce the number of search domains in the list, among which phasal transfer or Phase Impenetrability Condition (PIC) being a critical and well accepted candidate.

[16]This type of search algorithm is considered in EKS (2020).

algorithm would find *both* the subject and the object.

(6) $\{\mathrm{T}_{[u\phi]}, \{_{vP}\ \{_{DP}\ \mathrm{the}_{[v\phi]}\ \mathrm{girl}\}, \{_{v'}\ \mathrm{read}, \{\mathrm{the}_{[v\phi]}\ \mathrm{book}\}\}\}\}$

This is because the searches into DP and v' are parallel searches and they do not communicate with each other in terms of their search results, according to the modular search algorithm. Consequently, after confirming that the D head of the subject *the girl* bears the search target and *read* does not bear the search target, the search into v' will continue to look for the search target. The search terminates only after v' and all accessible subsets of v' are exhaustively visited and the head of the object *the book* is returned.

Such a result is not what we want. Note that the hierarchical nature of language is one of language's most crucial characteristics. The subject c-commands the object; however, the modular algorithm returns both the subject and the object (more accurately their heads), thus ignoring one of the most important features of language. Empirically, if we adopt the modular algorithm, we must have a way to prevent Minimal Search from returning the object. One may say that the object is not available for Minimal Search for some special reason (e.g., it has been assigned case or it has been transferred when T enters the derivation[17]). But we know from other languages that the object must be generally available for Minimal Search as long as the subject does not bear the search target. This has been clearly shown in the following contrast in Icelandic (taken from Sigurðsson 1996):

(7) a.

Við	lásum/*las	bókina.
we.NOM	read.**1PL**/*3SG	the-book.ACC

‘We read the book.’

b.

Henni	leiddust	strákarnir.
her.DAT	bored.**3PL**	the-boys.NOM.PL

‘She found the boys boring.’

When the subject bears the nominative case, T agrees with it (7a). However, when the subject bears the “inherent” dative case, T does not agree with the subject but can agree with the object which bears the nominative case (7b).

[17]Note that according to Chomsky (2015b) the internal argument is not transferred when T enters the derivation, only its lower copy is.

Cyclic Agree is another case where the object is available only when the subject is not available for Agree. Béjar and Rezac (2003), Rezac (2004), and Béjar and Rezac (2009) argue that agreement in Georgian occurs in two cycles. In the first cycle, a head with unvalued features (the trigger) probes downward to find a head matching the feature attributes of the unvalued features. If the search in the first cycle is not successful, the trigger may instead search upward. Béjar (2003) and Béjar and Rezac (2003) assume that person and number features are represented separately in Georgian: *v* bears a [uPerson] feature and T bears a [uNum] feature. (8a) shows that, when the subject is plural, $\text{T}_{\text{[uNum]}}$ agrees with the subject and $v_{\text{[uPerson]}}$ agrees with the object.

(8) Georgian (Rezac 2004, p. 72)

a. m-xedav-t
1-see-PL
'You(.PL) see(.PL) me.'

b. g-xedav-t
2-see-PL
'I see(.PL) you(.PL).'

By contrast, (8b) exemplifies that, when the subject is singular, $v_{\text{[uPerson]}}$ still agrees with the object, but $\text{T}_{\text{[uNum]}}$ now agrees with the object instead of the subject. Béjar and Rezac suggest that the singular number feature on the subject is underspecified or not present in syntactic representations in Georgian, so downward searching from $\text{T}_{\text{[uNum]}}$ skips the subject and expands the search domain down to the object. The contrast between (8a) and (8b) strongly support the idea that if two syntactic objects α and β both bear the search target and if α asymmetrically c-commands β, α rather than β will be found by Minimal Search for agreement purposes. Only when α is not accessible for Minimal Search could β then be returned. The global search algorithm for Minimal Search thus yields correct predictions. This is in fact resonant with Chomsky's (1993) Shortest Movement Condition, his (1995) Minimal Link Condition, and Rizzi's (1990) Relativized Minimality. By contrast, the modular algorithm makes wrong predictions as it allows both α and β be returned even if α asymmetrically c-commands β.

3. Minimal search-based Agree and labeling

Above, I gave a formal definition of Minimal Search. In this section, we will give a Minimal Search-based definition for labeling and another for Agree. It will become clear that Agree and labeling can be partially but not completely reduced to Minimal Search.

As shown in (5), Minimal Search involves three components, the SD, ST, and SA. The SA is invariant and identical for all cases of Minimal Search, no matter whether it is for Agree or labeling. What distinguishes Agree from labeling is the SD and ST. The SD and ST being independently assigned by different operations is a unique feature of the current definition of Minimal Search which is not available previously (see Ke (2019) ch. 2 for an extended exploration of the empirical implications of this feature). Below I would like to explain how the SD and ST are determined in Agree and labeling.

3.1 How to determine SD and ST?

The SDs of Minimal Search for labeling and Agree are determined in accordance with the purpose of these two operations. Agree connects two heads, one carrying unvalued features and the other carrying matching valued features. In order for two heads in an agreement relation to be connected to each other syntactically, they must form a syntactic relation at a certain stage of derivation. That is, they must be contained in the same root set or under a single node in a syntactic tree at a certain stage of derivation. Suppose a head X with an unvalued feature uF, represented as $X_{[uF]}$, needs to be valued by another head Y with a corresponding valued feature vF, represented as $Y_{[vF]}$, there are three logically possible situations for X to be connected to Y at a certain point of the derivation, with regard to the c-command relations between X and Y:

(9) a. $X_{[uF]}$ merges with $Y_{[vF]}$ or a set containing $Y_{[vF]}$: $X_{[uF]}$ c-commands $Y_{[vF]}$;

b. $Y_{[vF]}$ merges with $X_{[uF]}$ or a set containing $X_{[uF]}$: $Y_{[vF]}$ c-commands $X_{[uF]}$;

c. A set α containing $Y_{[vF]}$ merges with a set β containing $X_{[uF]}$. Neither $X_{[uF]}$ nor $Y_{[vF]}$ c-commands the other.

All three options are utilized in natural language. Downward (long-distance) Agree, e.g., where the matrix verb agrees with an embedded nominal for noun classes in Tsez (Polinsky and Potsdam 2001) is an instance of (9a). Upward Agree in negative concord (Zeijlstra 2012), inflection doubling/parasitic participles (Wurmbrand 2012) and multiple case licensing (Hiraiwa 2001) are instances of (9b). And Ke (2019, ch. 3) argues that reflexive binding is an instance of (9c). We will set (9c) aside for the discussion below.

Importantly, with (9a), the Minimal Search initiated to value $X_{[uF]}$ needs to search into the c-command domain of X. That is, the SD of the Minimal Search in this case must be the sister, or co-member, of X. However, a problem immediately arises with regard to (9b), where the valuee bearing the unvalued feature, $X_{[uF]}$, is c-commanded by the valuer bearing the corresponding valued feature, $Y_{[vF]}$. Notice that there is no way for the specific formalization of Minimal Search in (5) to search "upward." Minimal search as defined in (5) always searches "downward" into its SD. This problem can be resolved with a search domain assigned as a set that contains the c-commanding valuer. Due to space limitation, we will not discuss this case here (see Ke 2019 ch. 2 for a proposal that formalizes upward Agree as a special case of downward Minimal Search).

The ST for Agree is simply the features attributes of the unvalued features on the head that initiates the Minimal Search. For instance, when the purpose of a Minimal Search is to find a head with valued features matching the unvalued feature attributes [Person: , Num:] on a T head, then the feature attributes [Person: , Num:] are taken as the ST (see Footnote 4).[18]

On the other hand, since the labeling algorithm is meant to provide a label for a given set, the SD will be that set. For instance, if the set {kick, {the, ball}} is the set that needs to be labeled, this set is the SD of the Minimal Search for the labeling purpose.

The ST of Minimal Search in labeling is more complicated. Take (10) as an example. In the

[18]I leave open for now the question whether all the relevant feature attributes should be combined as a single ST and initiate a single Minimal Search, or each of them should initiate an independent Minimal Search. For most cases, these two options do not make a distinction. For exceptional cases where we may need to split the feature attributes, please see e.g., Béjar and Rezac (2003) and Preminger (2009). The current definition of Minimal Search is in principle compatible with the split analysis. A mixed analysis with a combined ST in some cases but with split STs in some other cases may also be possible.

case of (10a), the ST would be any feature or any head, not a particular head or feature.

(10) a. $\{_\alpha$ H, XP$\}$: $\{_\alpha$ kick, $\{$the ball$\}\}$

b. $\{_\beta$ XP, YP$\}$ = $\{_\beta$ $\{_\delta$ X, KP$\}$, $\{_\kappa$ Y, ZP$\}\}$:

$\{_\beta$ $\{_\delta$ a, $\{$little boy$\}\}$, $\{_\kappa$ is, $\{$kicking the ball$\}\}\}$

However, for (10b), if we want to obtain a $<\phi,\phi>$ pair by means of Minimal Search, then the ST must be ϕ-features as well, a set of specific features. This assumes that the labeling algorithms, STs specifically, for (10a) and (10b) should be different. A unified labeling algorithm, by contrast, has to either (i) set the ST to any feature/head (not particular features), and therefore will return only heads for both (10a) and (10b); or (ii) set the ST to some specific features such as ϕ-features. A problem for (ii) is that when features other than ϕ-features are in agreement, e.g., in an interrogative CP, the features in agreement, i.e., the <Q, Q> pair, cannot be directly returned as a label, unless the ST of the Minimal Search for an interrogative CP is set to Q. Furthermore, it is important to point out that even if we want to set different STs for different searches depending on the type of prominent features in $\{$XP, YP$\}$ constructions, we have a look-ahead problem: how do we know what type of prominent features are relevant and thus set these features as STs before searching into the search domain? In order to avoid the look-ahead problem and have a unified labeling algorithm for all the cases we discussed (we surely do!), the ST should be set to any head/any feature in all cases, and thus the label for (10b) must be <a, is> rather than a $<\phi,\phi>$ pair for an $\{$XP, YP$\}$ construction such as (10b) (see more discussion below in Section 4.2).

To summarize, I have shown briefly that the *search algorithm* in Agree and labeling can be unified by the search algorithm of Minimal Search. However, the *values* of the SD and ST of Minimal Search are independently determined by Agree and labeling, and thus the SD and ST values for Agree are different from those for labeling. Minimal search for Agree and labeling thus serve distinct purposes. More details of this argument will be presented later in Section 4.

3.2 MS-Agree and MS-labeling

We are now ready to give definitions of Minimal Search-based Agree and Minimal Search-based labeling, which I will henceforth abbreviate as MS-Agree and MS-labeling:

(11) Definition: Minimal Search-based Agree (MS-Agree)

Agree = Minimal Search + Valuation

i. **Minimal search**

a. Input to Minimal Search:

- SD = co-member of $\mathrm{H}_{[\mathrm{uF}]}$
- ST = feature attributes of uFs (e.g., [Person: , Num:])

b. Output of MS: heads bearing the ST

ii. **Valuation**

- Trigger: the head bearing uFs
- Valuation: copy the value of the ST in the Minimal Search output to the corresponding uFs in the trigger

Below, I give an example to illustrate the implementation of MS-Agree. *Be* in (12) can agree with *a man*, or more accurately, the determiner *a*, the head of *a man*. This is a case of long-distance agreement. The steps of MS-Agree are shown in (13).

(12) Example: *(there) be likely to be a man outside*

(13) a. SD = {likely, {to be a man outside}}

b. ST = ϕ-feature attributes of the unvalued ϕ-features on *be*

c. Search into the SD, the target is not found

d. Set {to, {be a man outside}} as the SD

e. Search into the SD, the target is not found

f. Set {be, {a man outside}} as the SD

g. Search into the SD, the target is not found

h. Set {{a man}, outside} as the SD

i. Search into the SD, the target is not found

j. Set {a, {man}} as the SD

k. Search into the SD, the target is found on *a*, return *a*

l. Valuation: copy feature values to *be* from the matching features on *a*

Furthermore, a formal definition for MS-labeling is presented below:

(14) Definition: Minimal Search-based labeling (MS-labeling)

The output of Minimal Search is taken as the label of the set to be labeled.

i. Input to Minimal Search:

- SD = the set to be labeled
- ST = any feature/head[19]

ii. Output of Minimal Search:

- Heads

The labeling algorithm is straightforward. Let us use the examples in (1), repeated below in (15), as an illustration.

(15) a. $\{_\alpha$ kick, {the ball}} $\longrightarrow$ label = kick

b. $\{_\beta$ $\{_\delta$ a, {little boy}}, $\{_\kappa$ is, {kicking the ball}}} $\longrightarrow$ label = {a, is}

What is different from the standard theory of labeling is that now we have <a, is> rather than <ϕ, ϕ> as the label of the structure in (15b). I will return to the implications of this difference later.

4. Implications of MS-Agree and MS-labeling

4.1 Agree and labeling can only be partially reduced to Minimal Search

Recall that one of the primary motivations for giving a formal definition of Minimal Search is to evaluate the appealing conjecture proposed by Chomsky (2013, 2015a): both Agree and labeling can be reduced to Minimal Search.

[19]This is equal to the statement that ST is not a set and ST is not empty.

With the formal definition of Minimal Search in (5) and the definitions of MS-Agree and MS-labeling in (11) and (14), it is quite clear that Agree and labeling cannot be completely reduced to Minimal Search. The main reason is that the specific *values* of SDs and STs for Minimal Search are determined by Agree and labeling independently. The assignment of the values of SDs and STs thus serves different purposes in Agree or labeling. Therefore, even if we take the Valuation step in Agree as a separate operation that occurs at SM/PF and not as part of the syntax of Agree, so that both labeling and the syntactic part of Agree are essentially Minimal Search, we still cannot reduce Agree and labeling fully to Minimal Search, given that the SDs and STs are assigned by Agree and labeling separately.

In addition, the definition of MS-Agree in (11) shows that although Agree uses Minimal Search to find the target, we must resort to the second step, Valuation, in order to copy the values of the valued features from the target to the corresponding unvalued features on the trigger. The definition of MS-labeling in (14), on the other hand, is essentially Minimal Search. But still labeling is an independent linguistic operation that takes the output of Minimal Search as the label. In other words, labeling specifies a way to use the output of Minimal Search. Minimal search itself does not give instructions or put any restrictions on how to use its search output, as the output could feed either Agree or labeling.

It is worth noting that with Minimal Search, we do unify a core component of Agree and labeling. The search algorithms for Agree and labeling is the Minimal Search algorithm. This is partially in accordance with Chomsky's intuition regarding a possible reduction of Agree and labeling to Minimal Search.

Therefore, I conclude that although Agree and labeling both involve Minimal Search, they are linguistic mechanisms that are independently needed, arguably for interface interpretation purposes. If we take Agree and labeling away from Syntax, Minimal Search as a general search algorithm will not know for what and into which domain it should search. This is probably a general mechanism underlying the application of 3rd factor principles in language: 3rd factor principles serve general purposes, and their specific application in language must interact with linguistic operations for them to accomplish particular linguistic purposes. In the case of MS-Agree and MS-labeling, the input to Minimal Search is provided by specific linguistic operations (Agree or labeling), whereas the output of Minimal Search is handled by these linguistic operations.

4.2 A complication with <ϕ, ϕ>

Furthermore, as we have mentioned, if we have a simple, unified labeling algorithm, we could not on the one hand obtain a head as the label for instances such as {kick, {the, ball}}, and on the other hand return a ϕ-feature pair for instances such as {{a, {little boy}}, {is, {kicking the ball}}} (see (10)). One way to return a ϕ-feature pair in the second case is setting the ST to ϕ-features (see Chomsky 2013, p. 13). In fact, according to this approach, for any type of feature pairs such as <ϕ, ϕ> and <Q, Q>, the search algorithm needs to set that type of feature attributes as the ST. In other words, the search algorithm has to be assigned an ST whose content depends on the specific type of the SD. This is a look-ahead property that is not desirable, and I believe it cannot be easily implemented by a simple search algorithm. A simple search algorithm that can avoid this look-ahead problem is that Minimal Search always looks for some specific types of features that can identify syntactic objects. The type of features that is most suitable for labels is of course categorial features. However, if Minimal Search always searches for categorial features, it cannot look for ϕ-features or Q-features. Consequently, this algorithm turns out to be almost identical to the one advocated in this paper.

A potentially feasible way to return a <ϕ, ϕ> pair from {{a, {little boy}}, {is, {kicking the ball}}} is to add another operation to MS-labeling: after *a* and *is* are returned by Minimal Search, MS-labeling compares their features and returns their shared features as a feature pair. The feature pair is then taken as the label. In terms of the visual search task in Figure 1 above, an analogue would be that we are searching for (any) animals first, if the task is searching for animals. Once we find two animals, we compare their features. If both of them are hairy, we can use the shared feature <hairy, hairy>, or “hairy things,” to label them. This complicates the labeling algorithm as it adds an additional comparison operation. This additional operation is vacuous for cases such as {kick, {the, ball}}, since only one head *kick* will be returned. An interesting question to ask here is whether this additional operation is theoretically and empirically preferable. Unfortunately, due to space limitations, I will not discuss this issue here any further.

4.3 Other potential implications

There are other implications of the definition of Minimal Search and MS-Agree and MS-labeling that we cannot discuss here due to space restrictions. For instance, based on the definition of MS-Agree in (11), it can be naturally derived that Agree consists of two sub-operations: Minimal Search and Valuation, consistent with proposals found in the literature (Arregi and Nevins 2012; Bhatt and Walkow 2013; Smith 2017; Kučerová 2018; Bjorkman and Zeijlstra 2019). The separation is natural under the current proposal because Minimal Search does not in any way tell us how to copy the valued features on the returned head to the unvalued features that have initiated the search.

5. Conclusions

This paper proposes a formal definition of Minimal Search to evaluate the idea that both Agree and labeling can be reduced to Minimal Search. Different aspects of the search algorithm, e.g., breadth-first vs. depth-first search, parallel vs. serial search, global vs. modular search are discussed, and reasons for making choices between each of these pairs are given based on detailed examinations of their theoretical and empirical consequences. Furthermore, MS-Agree and MS-labeling are defined based on the definition of Minimal Search. I have shown that MS-Agree and MS-labeling are clearly two distinct operations: although they share the same search algorithm, they serve different purposes and involve different values of search domain and search target. In addition, Agree consists of two sub-operations: Minimal Search and Valuation, whereas labeling does not share the Valuation operation. Finally, the outputs of Minimal Search in Agree and labeling are handled in different ways. Therefore, Agree and labeling can be only partially but not completely reduced to Minimal Search.

6. Appendix: Pseudo-code for the SA in Minimal Search

Note that in the pseudo-code I obtain the results of the parallel Minimal Search algorithm in (5) by applying the Multi_Find function, which aggregates a list of returned heads by executing the Find function in multiple processors simultaneously (e.g., the *multiprocessing* package in Python provides such an environment), assuming that multiple processors are available for Minimal Search.

```
GLOBAL VARIABLE:
    new_SD = List()
FUNCTIONS:
    1. List($x$, $y$):
        return: [$x$, $y$]
    2. Member($x$={$y$, $z$}):
        return: List($y$, $z$)
    3. Multi_Find($x$, $y$=[$a$, $b$, $c$...]):
        return:
            List(Multi_processor(Find($x$, $a$), Find($x$, $b$), Find($x$, $c$), ...))
    4. Find($x$, $y$):
        found_heads = List()
        if $y$ is a head and $y$ includes $x$:
            append $y$ to found_heads;
        if $y$ is a set:
            join new_SD and Member($y$)
        if $y$ is a list:
            Multi_Find($x$, y):
        return: (found_heads, new_SD)

  ALGORITHM:
```

```
Start: Terminate_search = False, new_SD = List(), given ST, SD

While Terminate_search is equal to False:
        (found_heads, SD) = Find(ST, SD)
        if found_heads is not empty:
            Terminate_search = True
        else if SD is not empty:
            new_SD = List()
```

References

Arregi, Karlos, and Andrew Nevins. 2012. *Morphotactics: Basque auxiliaries and the structure of spellout*, volume 86. Springer.

Aycock, Seth. 2020. A third-factor account of locality: explaining impenetrability and intervention effects with minimal search. In *Cambridge occasional papers in linguistics*, volume 12(1), 1–30.

Béjar, Susana. 2003. Phi-syntax: A theory of agreement. Dissertation, University of Toronto.

Béjar, Susana, and Milan Rezac. 2003. Person licensing and the derivation of PCC effects. In *Romance linguistics: Theory and acquisition: Selected papers from the 32nd Linguistic Symposium on Romance Languages (LSRL)*, ed. Ana Teresa Pérez-Leroux and Yves Roberge, volume 4 of *Amsterdam Studies in the Theory and History of Linguistic Science*, 49–62. Amsterdam/Philadelphia: John Benjamins Publishing.

Béjar, Susana, and Milan Rezac. 2009. Cyclic agree. *Linguistic Inquiry* 40:35–73. URL `https://muse.jhu.edu/journals/linguistic_inquiry/v040/40.1.bejar.html`.

Bhatt, Rajesh, and Martin Walkow. 2013. Locating agreement in grammar: An argument from agreement in conjunctions. *Natural Language & Linguistic Theory* 31:951–1013.

Bjorkman, Bronwyn M, and Hedde Zeijlstra. 2019. Checking up on (ϕ-) agree. *Linguistic Inquiry* 50:527–569.

Branan, Kenyon, and Michael Yoshitaka Erlewine. to appear. Locality and (minimal) search. In *Cambridge handbook of minimalism*, ed. Kleanthes K. Grohmann and Evelina Leivada. Cambridge, UK: Cambridge University Press.

Chomsky, Noam. 1993. A minimalist program for linguistic theory. In *The view from building 20: Essays in linguistics in honor of sylvain bromberger*, ed. Kenneth Hale and Samuel Jay Keyser, Current Studies in Linguistics, book section 1, 1–52. Cambridge, MA: The MIT Press.

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

Chomsky, Noam. 2000. Minimalist inquiries: The framework. In *Step by step: Essays on minimalist syntax in honor of Howard Lasnik*, ed. Roger Martin, David Michaels, and Juan Uriagereka, 89–155. Cambridge, MA: The MIT Press.

Chomsky, Noam. 2004. Beyond explanatory adequacy. In *Structures and beyond*, ed. Adriana Belletti, volume 3, 104–131. Oxford: Oxford University Press.

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

Chomsky, Noam. 2008. On phases. In *Foundational issues in linguistic theory: Essays in honor of Jean-Roger Vergnaud*, ed. Robert Freidin, Carlos P. Otero, and Maria Luisa Zubizarreta, 133–166. Cambridge, MA: The MIT Press.

Chomsky, Noam. 2013. Problems of projection. *Lingua* 130:33–49.

Chomsky, Noam. 2015a. A discussion with Naoki Fukui and Mihoko Zushi (march 4, 2014). *Sophia Linguistica: Working Papers in Linguistics* 69–97. URL `http://digital-archives.sophia.ac.jp/repository/view/repository/00000036046`.

Chomsky, Noam. 2015b. Problems of projection: Extensions. In *Structures, strategies and beyond : Studies in honour of Adriana Belletti*, ed. Elisa Di Domenico, Cornelia Hamann, and Simona Matteini, volume 223, book section 1, 3–16. Amsterdam/Philadelphia: John Benjamins Publishing Company.

Chomsky, Noam. 2019a. MIT lectures. URL `https://youtu.be/r514RhgISv0https://youtu.be/GPHew_smDjY`.

Chomsky, Noam. 2019b. UCLA lectures. URL `https://linguistics.ucla.edu/noam-chomsky/`.

Epstein, Samuel D, Hisatsugu Kitahara, and T Daniel Seely. 2020. Unifying labeling under minimal search in ”single-” and ”multiple-specifier” configurations. In *Coyote papers: Working papers in linguistics, linguistic theory at the university of arizona*. University of Arizona Linguistics Circle. URL `http://hdl.handle.net/10150/641481`.

Epstein, Samuel David, Hisatsugu Kitahara, and T Daniel Seely. 2014. Labeling by minimal search: Implications for successive-cyclic a-movement and the conception of the postulate “phase”. *Linguistic Inquiry* 45:463–481. URL `https://muse.jhu.edu/journals/linguistic_inquiry/v045/45.3.epstein.html`.

Fodor, Jerry. 1983. *The modularity of mind*. Cambridge, MA: The MIT Press.

Hiraiwa, Ken. 2001. Multiple agree and the defective intervention constraint in Japanese. In *MIT Working Papers in Linguistics*, volume 40, 67–80. Cambridge, MA: MITWPL.

Horowitz, Todd S, and Jeremy M Wolfe. 1998. Visual search has no memory. *Nature* 394:575–577.

Ke, Hezao. 2019. The syntax, semantics and processing of agreement and binding grammatical illusions. Dissertation, University of Michigan, Ann Arbor.

Krivochen, Diego. 2021. The search for minimal search URL `https://ling.auf.net/lingbuzz/005817`, manuscript.

Kučerová, Ivona. 2018. ϕ-features at the syntax-semantics interface: Evidence from nominal inflection. *Linguistic Inquiry* 1–77. URL `https://doi.org/10.1162/ling_a_00290`.

Larson, Bradley. 2015. Minimal search as a restriction on merge. *Lingua* 156:57–69. URL `http://www.sciencedirect.com/science/article/pii/S0024384114002939`.

MacDonald, Maryellen C., Neal J. Pearlmutter, and Mark S. Seidenberg. 1994. The lexical nature of syntactic ambiguity resolution. *Psychological Review* 101:676–703.

Marr, David. 1982. *Vision: A computational investigation into the human representation and processing of visual information*. The MIT Press.

Milway, Daniel. 2021. Agree as derivational operation: Its definition and discontents URL `https://ling.auf.net/lingbuzz/005842`, manuscript.

Moro, Andrea. 1997. Dynamic antisymmetry: Movement as a symmetry-breaking phenomenon. *Studia Linguistica* 51:50–76. URL `http://dx.doi.org/10.1111/1467-9582.00017`.

Ott, Dennis. 2011. Local instability: The syntax of split topics. Dissertation, Harvard University.

Polinsky, Maria, and Eric Potsdam. 2001. Long-distance agreement and topic in Tsez. *Natural Language & Linguistic Theory* 19:583–646.

Preminger, Omer. 2009. Breaking agreements: Distinguishing agreement and clitic doubling by their failures. *Linguistic Inquiry* 40:619–666.

Pylyshyn, Zenon W, and Ron W Storm. 1988. Tracking multiple independent targets: Evidence for a parallel tracking mechanism. *Spatial vision* 3:179–197.

Rezac, Milan. 2004. Elements of cyclic syntax: Agree and merge. Dissertation, University of Toronto.

Richards, Marc. 2007. On feature inheritance: An argument from the phase impenetrability condition. *Linguistic Inquiry* 38:563–572.

Rizzi, Luigi. 1990. *Relativized minimality*. Linguistic inquiry monographs 16. Cambridge, MA, US: The MIT Press.

Sigurðsson, Einar Freyr. 2017. Deriving case, agreement and voice phenomena in syntax. Dissertation, University of Pennsylvania.

Sigurðsson, Halldór Ármann. 1996. Icelandic finite verb agreement. In *Working Papers in Scandinavian Syntax*, volume 57, 1–46. Department of Scandinavian Languages, Lund University.

Smith, Peter W. 2017. The syntax of semantic agreement in English. *Journal of Linguistics* 53:823–863.

Trueswell, John C, Michael K Tanenhaus, and Susan M Garnsey. 1994. Semantic influences on parsing: Use of thematic role information in syntactic ambiguity resolution. *Journal of memory and language* 33:285–318.

Trueswell, John C., Michael K. Tanenhaus, and Christopher Kello. 1993. Verb-specific constraints in sentence processing: Separating effects of lexical preference from garden-paths. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 19:528–553.

Wolfe, Jeremy M. 2010. Visual search. *Current Biology* 20:R346—R349.

Woodman, Geoffrey F, Edward K Vogel, and Steven J Luck. 2001. Visual search remains efficient when visual working memory is full. *Psychological Science* 12:219–224.

Wurmbrand, Susi. 2012. Parasitic participles in germanic: Evidence for the theory of verb clusters. *Taal en Tongval* 64:129–156.

Zeijlstra, Hedde. 2012. There is only one way to agree. *The Linguistic Review* 29:491–539. URL `https://www.degruyter.com/view/j/tlr.2012.29.issue-3/tlr-2012-0017/tlr-2012-0017.xml`.