The search for Minimal Search

Diego Gabriel Krivochen
diegokrivochen@hotmail.com

Abstract

In this paper we examine Minimal Search, an operation that is at the core of current Minimalist inquiry. We argue that, given Minimalist assumptions about structure building consisting of set-formation, it is not possible to define Minimal Search as a search algorithm, which would partly explain the lack of explicit formal characterisations of this operation in the literature. Furthermore, some problematic configurations for Minimal Search (namely, {XP, YP} and {X, Y}) are argued to be an artefact of these set-theoretic commitments. However, if set-formation is given up as the core operation in syntax in favour of directed graphs, Minimal Search can be straightforwardly characterised as a search algorithm that applies unambiguously. Its usefulness for syntactic analysis may thus go beyond the role it plays in Minimalism.

Keywords: set theory; graph theory; Minimal Search; phrase structure

1. Introduction

Minimal Search (MS henceforth) was introduced under that name in the context of the Problems of Projection program, as a 'third factor' property (Chomsky, 2013: 43) which '*falls under MC* [Minimal Computation]' (Chomsky, 2015: 6). Similar characterisations of Minimal Search (MS) are to be found e.g. in Epstein et al. (2017, 2020), Chomsky (2020a: 36, 2020b), Goto (2019), Komachi et al. (2019), Van Gelderen (2019), Larson (2015), Hayashi (2021). MS has become a crucial aspect of Minimalist theorising, as a key component of operations like Labelling and Agree: for example, Bauke & Blühmel (2017: 4) refer to MS in the context of a presentation of the so-called Labelling Algorithm, echoing Chomsky in saying that

> *the labelling algorithm based on the notion Minimal Search that implements this requirement in a computationally efficient manner*

If Labelling is a properly defined algorithm, as an unambiguous sequence of steps that carries out a computational process which maps an input to an output of specific kinds (in this case, labelling would be a function from syntactic objects into labels), then MS being such a central component should also be properly defined. Similarly, Collins (2017) appeals to MS in a definition of linearisation procedures under a strong set-theoretical commitment: Merge(X, Y) = {X, Y}. Chomsky (2013, 2015, 2020a, b) resorts to MS in the discussion about subject movement (from Spec-*v*P to Spec-TP, seeking to eliminate the EPP), labelling, accessibility in successive cyclic movement

(2020a: 36), and possibly other processes. In this context, is it surprising that there is no formal definition of MS to be found (Collins & Stabler's 2016 formalisation of Minimalist Syntax does not even mention MS). Characterisations in the literature are at best informal, as in Chomsky (2020a: 36):

> [MS is] *a third factor principle, that says look for the closest copy and don't look any further*

Or Vercauteren (2017: 70) (here, MS is again invoked in relation to Labelling):

> *the head that is closest to the node to be labeled will provide the label for the whole structure*

But most times, MS is simply assumed, without even being informally characterised. Concepts like 'distance' and 'computational efficiency' (or 'Minimal Computation') appear scattered in these informal characterisations, but are themselves not defined (see Lappin et al., 2001; Postal, 2004 for a detailed critique of similar formal imprecisions regarding 'perfection' and 'virtual conceptual necessity', notions independent from empirical concerns). It is not entirely difficult to see why: given contemporary Minimalism's axiomatic commitment to sets (such that binary set formation would be, by hypothesis, the 'simplest' possible operation; Chomsky, 2013: 42; 2020a: 22), the issue becomes defining distances in sets without having a mathematical framework to do so.
In current Minimalist theorising, MS is relegated to the realm of the 'third factor', which comprises

> *3. Principles not specific to the faculty of language.*
>
> *The third factor falls into several subtypes: (a) principles of data analysis that might be used in language acquisition and other domains; (b) principles of structural architecture and developmental constraints that enter into canalization, organic form, and action over a wide range, including principles of efficient computation, which would be expected to be of particular significance for computational systems such as language.* (Chomsky, 2005: 6)

These characterisations give MS an *ex machina* flavour, which is -needless to say- undesirable. What can be done, then? Is there a way to define MS in the context of contemporary Minimalism in a way that is both formally explicit and empirically fruitful? Here we will argue that MS cannot be formally defined as a search algorithm under current Minimalist assumptions, and that it is the very nature of the computational system in the most recent incarnation of generative grammar that prevents an appropriate definition from being possible. In other words, there can be no 'search' in MS under Minimalist assumptions. A proper characterisation of MS requires a departure from some core tenets of Minimalism with respect to the format of structural descriptions and the properties that a search algorithm has.

Let us begin with some general considerations about MS and the framework in which it is currently used. 'Minimal' suggests that the search space should be bounded, stopping at the closest element that satisfies whatever requirement is involved in the 'Search': specifically, Chomsky (2013: 43) suggests that MS stops when it finds a head; Chomsky (2020a: 48) establishes that it stops once it reaches the head of a chain (which may or may not be a head in the sense of 'terminal node'). We will come back to the issue of deciding what counts as 'minimal' below. Now, we begin with the 'search': we need to ask how exactly this 'search' would take place and how we can provide a formal definition of this process. To address this question, we will introduce search algorithms and some crucial differences in the formal relations that can be defined under different approaches to what the generative procedure generates in **Section 2**. In order to evaluate the applicability of search algorithms to syntactic representations, we need to consider what kinds of objects the search would operate over. This will be the topic of **Section 3**.

We need to make some preliminary considerations, to be expanded on in **Section 2**. Contemporary Minimalist theory assumes (by axiom) that the generative procedure produces *sets*: Merge / MERGE are instances of an allegedly irreducible operation of set formation over elements in a workspace (Chomsky, 2019, 2020a, b). In set-theoretic syntax, the structural description for a sentence like *The man falls* is as in (1) (taken from Collins, 2017: 65):

1) {{the, man}, {{φ, T}, {fall, {the man}}}}

Note the use of unordered set notation: the sets produced by Merge / MERGE are not ordered: this is an important point to which we will return. To Chomsky, order (which is usually conflated with *linear precedence*) is not part of the computational system. This is a consequence of adopting *sets* as a model of syntax, and a crucial ingredient of our argument: it is of course possible to define an order over a set that is not precedence. For example, McCawley's (1968) graph-based approach (which interprets phrase structure rules as admissibility conditions for local trees) defines two distinct two-place relations: *precedes* and *node dominates*. The former pertains to linear order in a string, the latter, to relations in structural descriptions. Similarly, some versions of Dependency Grammars (e.g., Kahane & Lareau, 2016) and Metagraph Grammar (Postal, 2010: 26) formulate linear precedence statements that do not impact on structural relations; these frameworks have in common the focus on graphs rather than on sets. The distinction between set-based and graph-based syntax will become very relevant shortly.

1.1 What is a search algorithm?

Before addressing any issues pertaining to Agree, Move, etc., we need to establish *what it means to 'search'* in a formal context. In this work we will pursue the possibility that MS is actually intended

as a search algorithm. What is, then, a search algorithm? In computer science, a search algorithm is a sequence of well-defined, implementable instructions that retrieves some information stored in a data structure; in other words, a sequence of steps to locate a memory address and retrieve the information contained in that address (Sedgewick & Wayne, 2017; Knuth, 1998; Stephen, 1994). Simplifying a bit, a search algorithm probes into a sequence of 'addresses' or 'keys', where it looks for a target value. If the target value is found after a finite number of comparisons between the target value and values in the input data, the search is successful. This process can be expressed in terms of string searches, such that given a string *s* we can look for the first occurrence of substring *u* of arbitrary length (shorter than *s*).

A simple search algorithm can be exemplified as follows: suppose we have a set of values $\{V_1, \ldots V_n\}$, each of which is assigned a *key* (an address that allows us to retrieve the value) $\{K_1, \ldots K_n\}$. Then, a search algorithm may be called upon to find a specific K_x. The algorithm starts at K_i, checks whether $K_i = K_x$, and if it is, the procedure terminates. If not, it proceeds to K_{i+1} and checks again; the procedure is repeated until reaching K_n. This kind of search algorithm is called *sequential search*. Note that the algorithm knows *what* to find (namely, key K_x), and includes a procedure of comparison such that each key in the sequence is compared with the target value. Furthermore, the sequence is ordered, such that the search can proceed from K_i ($i = 1$) all the way to the end of the input sequence (K_n) and not leave anything unchecked. Variations of this algorithm are possible if the keys are themselves ordered in an increasing order (such that instead of $\{K_i, \ldots K_n\}$ we have $K_i < K_j < \ldots K_n$; this order may be alphabetical, numerical, etc.), if the algorithm is sensitive to the number of times a particular key has been accessed, etc. Most search algorithms are optimised such that backtracking is avoided if at all possible (this includes a preference for memory-less devices).

Search algorithms have been devised not only for table-ordered datasets (e.g., a phonebook), but also for datasets structured in tree form. *Prima facie*, it seems that we should focus on tree-search algorithms given the format of structural descriptions in generative grammar. However, this is misleading. As we will see in the next section, the Minimalist framework within which considerations of MS arise is strongly committed to the idea that structural descriptions are *sets*, not *graphs*. But we do need to introduce the basics of tree search algorithms, since it will help in making the case for the crucial differences between set-based syntax and graph-based syntax clearer, and the feasibility of MS in both. In order to do that, we must define some concepts from graph theory.

1.2 Graphs and sets

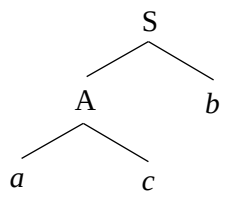
A graph is a pair G = (V, E), where V is a set of vertices (also called *nodes*) and E is a set of edges; $v \in V$ is a vertex, and $e \in E$ is an edge. An edge *e* joining vertices *a* and *b* is notated $e = \langle a, b \rangle$, and *a* and *b* are said to be *adjacent vertices*. The *neighbour set* of *v* is the set of adjacent vertices to *v*,

usually notated N(v), and the *degree* of v is the number of *edges* connected to it. For example, a vertex v with degree 2 and neighbourhood set 2(v) has two edges connected to it, and two vertices which are adjacent. The *indegree* of a vertex v_x is the number of edges that go *from* another vertex to v_x; the *outdegree* of v_x is the number of edges that go *from* v_x to some other vertex (Van Steen, 2010). There is not necessarily a correspondence between *degree* and *neighbourhood set*, since a vertex v_1 may be connected to v_2 by two distinct edges (in which case the neighbourhood set of v_1 would be 1 and its degree, 2). Let v_1 and v_2 be two (not necessarily distinct) vertices in G: a v_1-v_2 *walk* in G is a finite ordered alternating sequence of vertices and edges that begins in v_1 and ends in v_2. A *trail* is a walk without repeating edges, and a *path* is a walk without repeating vertices. Trees are, technically, specific kinds of graphs. A *tree* T is a graph that has no loops (there is no path in T that begins and ends in the same vertex) and is connected (for every two vertices v_x, v_y there is a finite path from v_x to v_y or vice-versa). Van Steen (2010: 113) puts it in the following terms:

> *A graph G is a tree if and only if there exists exactly one path between every two vertices u and v.*

The kinds of trees used as diagrams of sentence structure in generative grammar are, in addition, *simple* (no vertex can appear more than once), *directed* (such that edges have directionality; this defines the binary asymmetric relation *dominates* for every two adjacent vertices; a graph that contains directed edges is a *directed graph* or *digraph*), and *rooted* (there is a node that is not dominated by any other node). We can illustrate these notions. Consider the tree in (2):

2)



In (2), V = {*a*, *b*, *c*, A, S}. Nodes A and S have degree 2, whereas nodes *a*, *c*, and *b* have degree 1. S is the *root* of the graph.

In X-bar-style trees, heads always have degree 1, and non-heads always have degree 2: there is no unary branching or *n*-ary branching (for $n > 2$); this has been a staple of Merge since its inception, and continues to be an axiom of structure building under MERGE (Chomsky, 2020a: 22).

A set theoretic representation of (2) would be (3):

3) {S, {*b*, {A, {*a*, *c*}}}}

In (3), A contains {a, c} and is a subset of S, which contains {b} and the set {A, {a, c}}; in graph-theoretic terms A is a sub-graph of S *iff* V(A) ⊂ V(S) and E(A) ⊂ E(S). From this perspective, both

approaches allow us to capture the same relation. In graph-theoretic terms, we define the *root* of a directed tree as a designated node that is not dominated by any other node (Wilson, 1996: 56) or - more relevantly in the present context- in terms of the definition of paths in G:

> *A digraph* [directed graph] *G(V, E) is said to have a root τ if τ ∈ V and every vertex v ∈ V is reachable from τ; that is, there is a directed path that starts in τ and ends in v* [for all *v*]. (Even & Even, 2012: 37)

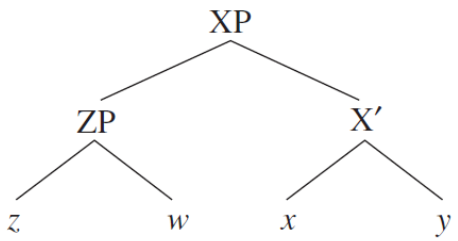
This notion has a set-theoretic analogue in syntactic theory:

> *K is the root iff:*
>
> *for any Z, Z a term of K, every object that Z is a term of is a term of K.* (Epstein et al., 2012: 262)

Chomsky (1995: 226) offers a direct translation between a tree diagram (note: not necessarily a *tree* in the graph-theoretic sense) and its set-theoretic interpretation:

> *Suppose that we have the structure represented informally as (9), with x, y, z, w terminals*

(9)



> *Here ZP = {z, {z , w}}, X' = {x , {x , y}}, XP = {x , {ZP, X'}}; more accurately, the tree with ZP as root corresponds to {z , {z , w }}, and so on, the labels of the roots having no status, unlike standard phrase markers. Note that w and y are both minimal and maximal; z and x are minimal only.*

In the set-theoretic representation, the status of ZP, X', and XP is unclear: they seem to be used as proxies for subsets, as informal ways to 'refer to' sets (Chomsky says they have 'no status', but then it is not clear why they are used at all). The 'name' of the set, in other words. But, as per *bare phrase structure* (Chomsky, 1994), they are not part of the syntactic representation *stricto sensu*. This is a very important difference with a graph-theoretic interpretation of a diagram like Chomsky's (9): in a graph, every node counts since it is part of the formal definition of a graph.

Chomsky's fragment may lead some readers to believe that sets and graphs are always equivalent, and that the choice between sets and graphs as the format of linguistic descriptions is merely notational. However, this is not the case. A crucial issue is there are relations that we can define in objects like (2) (graphs) but not in objects like (3) (sets); if these relations are relevant for the definition of MS,

then a set-theoretic stance runs into trouble given the centrality of MS in the definition of other syntactic processes. For example, in (2) we can define a path P between nodes *b* and *c* as an ordered set of nodes and edges:

4) $P_{b, c}$ = <*e*<*b*, B>, *e*<S, A>, *e*<A, *c*>>

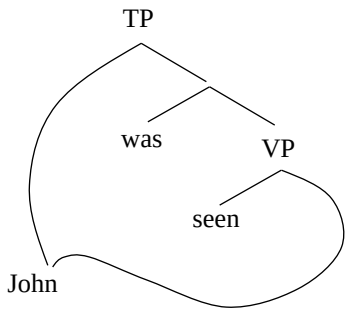
In (4) we defined an ordered set of directed edges: *e*<*x*, *y*>, for all *x*, *y*, is an edge *from x to y*. It is not possible, however, to define such a path for a pure set-theoretic representation, as in (3). Set-theory allows us to work in terms of membership (which in turn has been used to define the binary relation *term of*; Chomsky, 1995: 227; 2020a: 22). However, between members of an unordered set, it is not possible to define a sequence of terms in any way other than containment: this is the core of, for example, Kitahara's (2020) approach to MS, which we will come back to below. The possibility of defining paths will become fundamental for a proper definition of MS.

The fact that different relations can be defined in graphs and sets has led, in our opinion, to some confusion. For example, Collins & Groat (2018) argue against multidominance approaches (which allow any node to have an indegree greater than 1) by rejecting graph-theoretical approaches altogether based on the definition of Merge:

> *One issue that comes up right away is that* [a multidominance tree] *is a graph theoretic object. In minimalism, Merge forms sets {X, Y}*

The argument proceeds without considering the consequences of a graph-theoretic approach; rather, the authors focus on the problems that emerge with a set like {$John_1$ {T, {be, {seen, $John_1$}}}} as the structural description for the sentence *John was seen*. Collins & Groat correctly point out that the only way to make it work in Minimalism (where syntactic terminals are lexical tokens) is by introducing indices, thus violating the Inclusiveness Condition (a point also made in Krivochen, 2015). The reason indices are required is that otherwise we would be forced into the uncomfortable position of saying that {John} belongs to the set {seen, John} and doesn't at the same time. As we argued in Krivochen (2021), many problems related to the distinction between *copies* and *repetitions* arise precisely because of the set-theoretic commitments underlying Merge. In graph-theoretic terms, the structural representation for *John was seen*, assuming a more or less traditional generative tree, could go along the lines of (5)

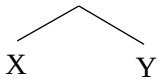
5)



If loops are allowed, then there is an edge from TP to *John* (which would correspond to Collins & Groat's leftmost 'John$_1$') *e*<TP, *John*> and an edge from VP to *John* (which would correspond to Collins & Groat's rightmost 'John$_1$') *e*<VP, *John*>. No diacritics are needed, because we can define two distinct directed edges that just so happen to converge at the same node; a graph-theoretic approach removes the necessity to have movement+indexing operations. The membership paradox that emerges in a pure set-theoretic approach dissolves under graph-theoretic assumptions (see also Krivochen, 2018b for an extensive presentation of empirical phenomena analysed in graph-theoretic terms). We have yet another example of the lack of direct translations between set-theoretic syntax and graph-theoretic syntax.

There are some issues that arises when trees are used to represent set-theoretic Merge (that is, when trees are used as diagrams of L-trees: the former being drawings and the latter being formal objects; see Postal, 2010: 7, ff.). Suppose that we Merge X and Y, yielding the set {X, Y}. How is that diagrammed? By using a binary-branching tree, as in (6):

6)



From a set-theoretic perspective, (6) may be seen as a graphical representation of the set {X, Y} (and it is assumed to be so in the Minimalist literature). However, from a graph-theoretic perspective we need to consider the fact that (6) has *three* nodes, not two. The tree in (5) is rooted, because there are *two edges*, each edge connecting two nodes. Since the edges converge, it means that there is a node (which we will call ●, remaining agnostic about the indexed category it is to be assigned to) such that e_1 <●, X> and e_2 <●, Y>. The formal object defined as the set {X, Y} and the graph in (5) are thus distinct, if both sets and graphs are taken seriously. A more accurate representation of Merge(X, Y) = {X, Y} would perhaps be (7) (see McKinney-Bock & Vergnaud, 2013: 219, ff. for a representation along these lines):

7) X ——— Y

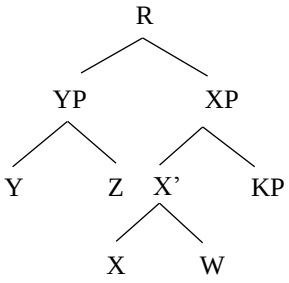
here no new nodes are introduced, we have just the input of Merge and an edge connecting the two terms involved in the operation (see Krivochen, 2018b: appendix A for extensive discussion). Trees and sets are not equivalent, and the choice between one and the other as the basis for syntactic theory has far-reaching consequences in terms of the relations and operations that can be defined in each. Much discussion pending, we intend this brief note to highlight the differences between set-theoretic and graph-theoretic approaches: this is important since part of our goal is evaluate the feasibility of MS as a search algorithm defined over sets formed by Merge/MERGE.

Above we introduced some fundamentals of search algorithms for sequences, now we need to do the same for trees. Tree search algorithms are broadly divided in two kinds: *breadth-first* and

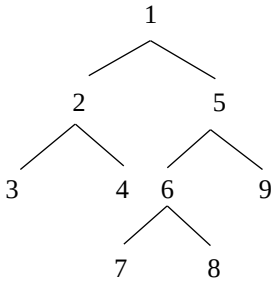
depth-first. Both have access to every node in a connected tree, but differ in terms of the order in which the search proceeds. A *breadth-first* search goes 'in waves' from the root, searching each generation from left to right (or right to left) before proceeding to the next generation. A node is marked as the start node, then all nodes adjacent to it are accessed one by one (and their values are added to a queue) until exhausting the set of nodes adjacent to the starting node. Then, the process continues at the next generation. A *depth-first* search also starts from the root (in a *preorder* transversal[1]), but instead of going through all the nodes in a generation before proceeding to the next, it explores the leftmost branch exhaustively before backtracking to the last branching parent node visited and proceeding with the immediately adjacent branch (Even & Even, 2012: 11, 46-49; Cormen et al. 2001: 531, ff.). We can illustrate the order in which nodes are visited in a simple tree for both search algorithms given a rooted, directed tree:

8)

a.

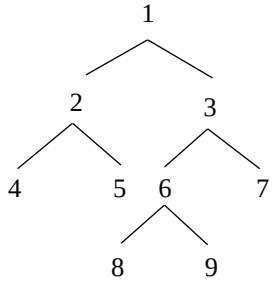


b.



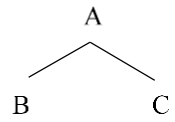
Depth-first

c.



Breadth-first

[1] Strictly speaking, there are three possible ways to implement a depth-first algorithm. If we take a simple branching node as an example:



A *preorder* transversal defines the sequence: A B C (root, left, right)
An *inorder* transversal defines the sequence: B A C (left, root, right)
A *postorder* transversal defines the sequence B C A (left, right, root)

The choice of *preorder* transversals in this paper obeys considerations of how the graph has been constructed and what MS is supposed to accomplish: for purposes of labelling, for instance, if we follow the generative literature, B and C are already labelled. The node that should be looking for a target is A, not B or C; thus, it makes sense to start from A. The same transversal would apply to a system where all predicates dominate their arguments, as in Dependency Grammars or the graph-theory based model in Krivochen (2018b).

In this context, we can define the length of the path between the root and any other node in the tree, as a sequence of nodes and edges. Suppose that the target value for the search algorithm is Y. A *depth-first* algorithm would define the following search sequence Σ:

9) Σ = *e*<R, YP>, *e*<YP, Y>

We can simplify (9) notationally, by indicating only the nodes in the order they are visited (omitting mention to edges, since we know already that any two adjacent nodes in a graph are linked by an edge), thus we get (10)

10) Σ = <R, YP, **Y**>

The algorithm would, at every node visited, compare the element in its input (the node) with its target value (a head). If the scanned symbol matches the target value, the algorithm halts; otherwise it keeps going. This mechanism underlies both depth-first and breadth-first algorithms, the only thing that changes is the order in which nodes are visited. For a breadth-first algorithm, the sequence would be:

11) Σ = <R, XP, YP, **Y**, Z, X', KP, X, W>

In this particular example, it just so happens that both algorithms find Y (bolded in (10) and (11)) before any other head; however, a depth-first algorithm finds Y after visiting two nodes, where as a breadth-first algorithm finds Y after visiting three. From the outside, we would prefer a depth-first since the target (which we know beforehand) is reached sooner. However, the argument does not carry over to the syntactic computation automatically: we have the luxury of looking at the structure as a whole, compare both methods, and choose exactly the one that finds an element that we can identify as a 'head' while defining the shortest path. But MS as a syntactic algorithm does not have access to the same kind of information we do: even recognising that something is a 'head' is far from a trivial task. To a great extent, this depends on the properties of the operation that generates the structure that the search algorithm is going to explore.

2. Merge: what it is and what it can do

It is necessary then to characterise the generative operation in order to evaluate the feasibility of defining a search algorithm for its output. Collins (2017: 50-53) gives a list of the properties of Merge: summarising,

- Merge is iterable (can apply to its own output)
- Merge is binary (the input of Merge is always a pair of objects)
- Merge is commutative (Merge(X, Y) = Merge(Y, X))
- (The output of) Merge is unspecified for linear order (see also Chomsky, 2013: 40; 2020b)

- (The output of) Merge is not assigned a label
- Merge is not triggered (by a head, a feature, etc.)
- Merge is never counter-cyclic
- Merge is all there is structure building-wise: there is no Move or Copy (just Merge + Agree)
- Merge cannot produce {XP, YP} or {X, Y} objects (where X and Y are heads)
- Merge allows (somehow[2]) to dispense with traces, indices, and copies
- Merge allows (somehow) to dispense with the notion of Chain

We will not argue for or against these properties, as it goes beyond the scope of this paper. They simply constitute the background against which the definition and role of MS can be evaluated. Collins' paper differs from Chomsky (2013, 2015, 2020a, b) in explicitly defining the object of study and its properties, which makes it a better candidate to begin our inquiry; the properties of Collins-style Merge that matter for our purposes also hold for Epstein et al.'s *Simplest Merge*[3]. Chomsky's recent works (2019, 2020a, b) propose a version of the generative engine called MERGE, which involves -as usual- set formation plus removal from a workspace. MERGE would apply as follows:

12) Workspace (WS) contains X and Y: [$_{WS}$ X, Y]
MERGE(X, Y) = [$_{WS}$ {X, Y} X, Y] (form a set containing X and Y)[4]
Remove X, Y from WS = [$_{WS}$ {X, Y}]

The basic properties of regular Merge are still there, in particular *binarity*. The removal of X and Y from the workspace intends to restrict the probing space and the number of elements available for further computations. Chomsky seems to equate 'size of the workspace' with 'cardinality of the set defined in the workspace', or, more simply, 'number of elements in the workspace'. This equation is

[2] Collins (2017: 53) claims that the non-existence of traces, indices, or copies (previously banned by the Inclusiveness Condition) follows from the definition of Merge:

> *there is no need to stipulate the inclusiveness condition as part of UG. Rather, it* ***follows as a theorem*** *from the definition of Merge.*

Exactly how, it is not made explicit. See Postal (2004: Chapter 9) for a critique of this kind of statement.

[3] Specifically, Epstein et al. (2015: 202) define *Simplest Merge* as follows:

> *1)* *Merge (α, β) → {α, β}*
>
> *Given (1), a syntactic object SO constructed from α and β by Merge is just {α, β}. Merge puts the two objects α and β into a relation, the output being represented as a two-membered set. Nothing more. Thus, unlike the output of Merge in Chomsky* [The Minimalist Program, Cambridge, Mass.: MIT Press] *and much subsequent work, the output of Merge in this conception does not overtly encode a label; contra Chomsky's* […] *notation, there is no constructed set {δ, {α, β }}, where δ represents the label (as either H(α) or H(β)) identifying the relevant properties of {δ, {α, β }}* […]. *Rather, under simplest Merge there is just {α, β}.*

[4] Strictly speaking, Chomsky (2020a, b) formulates the input of the operation as MERGE(X, Y, WS), as if WS was a syntactic object.

not innocent: the size of the workspace need not coincide with the *probing space* that an operation has access to within the workspace. A Turing machine has an infinite tape, but at any given time a single symbol can be read (see Krivochen, 2021 for extensive discussion about the formalisation of properties of the workspace).

According to Chomsky, if X and Y were not removed from WS, then we would have the set {X, Y}, the element X, and the element Y all accessible in the workspace (see the second line in (11)). Because accessibility in the workspace is equated to recursion (Chomsky, 2019: 280), the idea is that natural language's use of a *removal* operation sets NL recursion apart from 'general recursion' (Chomsky, 2019: 277-278; 2020a: 34)[5, 6]. Interestingly, Chomsky (2019: 279) rejects that the new system requires a *removal* operation (as the 1995 definition of Merge did), but rather MERGE *replaces* X and Y with {X, Y}. It is unclear how to formalise this, and if there is any formal substance in the use of *replace* instead of *remove*, but for our purposes it makes no difference. Chomsky (2020a) refers to so-called *third factor* properties and MS, our current focus.

This latest version of Minimalism differs quite substantially from previous stages of the theory. In the first incarnation of Minimalism, the generative operation Merge was defined as follows:

[5] There are numerous problems with this formulation, some of which we have analysed in depth in Krivochen (2021). One of them is that the notion of 'workspace' is left undefined. Another is that if X and Y in {X, Y} are not independently accessible, an operation of Copy becomes inescapable for instances of displacement, which in turn raises problems pertaining to copies vs. repetitions (Collins & Groat, 2018), the need to keep copies active for a number of derivational steps (copy something while it is still accessible, maintain it active somehow and somewhere, and then re-introduce it in the structural description; furthermore, you need a way to relate both copies in different phases) and more fundamentally a multiplication of entities beyond necessity. This is a consequence of a strong commitment to the idea that syntactic structure takes the form of sets (of sets). In this context, for cases of so-called *movement* (or, more theory-neutrally, *extraction*, as in Postal, 1998), copies and Internal Merge are unavoidable. This commitment is by no means the only possible choice. However, alternatives are not welcome:

> *there is a proposal, very widely used in the literature that we are talking about, which says we can overcome this by developing a new theory of movement, one which doesn't involve Internal Merge. So suppose we bar Internal Merge and develop a representational theory of movement, which just involves External Merge. That's very widely used, in multidimensionality. That's a terrible idea:* ***it violates every condition you can think of, and it has its own problems*** (Chomsky, 2019: 278)

Exactly what those problems are, which conditions are violated and how, and how the new theory of MERGE would account for the empirical insights from the approaches hereby rejected (multidominance, sidewards movement, non-transformational models) is not made explicit. It is doubtful, whether MERGE constitutes an instance of scientific progress (in the sense of Lakatos, 1978 and much related work; see also Lappin et al. 2001 for a similar point about some foundational Minimalist claims).

[6] *If MERGE is like normal general recursion (e.g. proof theory), then WS' will contain a and b as well as {a, b}. But as Chomsky points out, this makes it possible to generate illegitimate structures that violate well established linguistic constraints* (Fiengo, introduction to Chomsky, 2020a: vi)
What those linguistic constraints are specifically, and exactly how they would be violated, is not made explicit in currently available publications.

> *Applied to two objects α and β, Merge forms the new object K, eliminating α and β*. (Chomsky, 1995: 223)

Furthermore, due to the fact that syntactic objects are interpreted at the C-I and A-P interfaces differently depending on whether they are verbal, nominal, etc. (Chomsky, Op. Cit.), the object K was defined to be a set {γ, {α, β}}, with γ the label of {α, β}. Chomsky proceeds by saying that

> *K must therefore at least (and we assume at most) be of the form {γ, {α , β}}, where γ identifies the type to which K belongs, indicating its relevant properties.* (Op. Cit.)

The status of labels in the theory has changed greatly since the early days (where labels were simply nonterminal nodes in the sense of formal language theory), to the current situation where unlabelled objects are allowed (and in fact label-less objects seem to be desired) in syntactic derivations.
In any event, MERGE(X, Y) = {X, Y}. Overall, there seems to be nothing in Chomsky's recent papers and talks that constitutes a break with Collins' positions. Thus, we will refer to the outputs of generative operations, these unordered, unlabelled objects, as 'Chomsky-Collins sets'.
How do we know that the search is successful? This is possibly the only point that has been addresses explicitly: MS looks for a *head* bearing relevant features. If in the domain of the search a suitable head is found, other rules may apply (labelling, Agree, Internal Merge, etc.). For example, for purposes of labelling,

> *LA* [the Labelling Algorithm] *is trivial for {H, XP} structures, H a head. In this case, LA selects H and the usual operations apply* (Chomsky, 2015: 7)

But exactly how 'trivial' is it? In order for things to work the way Chomsky and others have suggested, it must be possible for the labelling algorithm, given a syntactic object, to determine whether there is a head: it means that the notion *head* needs to be hard-wired into the system. Suppose that we have the output of MERGE(A, B) = {A, B}. How do we know if either A or B is a head? If we are building sets, it matters greatly whether a syntactic terminal is a singleton or not. Suppose that we have

13) {*read*, {*the*, *book*}}

As the output of MERGE(*read*, {*the book*}). Do we want to consider *read* to be a unary set? This is important because if lexical terminals are singletons, then determining whether something is a head or not requires some additional operation apart from MERGE: for instance, the algorithm must be capable of evaluating the cardinality of a set. A head would be a set of cardinality 1, a non-head would be a set of cardinality greater than 1. At this point, Chomsky (2020a: 37) claims that

We want to say that [X], the workspace which is a set containing X is distinct from X.
[X] ≠ X
We don't want to identify a singleton set with its member. If we did, the workspace itself would be accessible to MERGE. However, in the case of the elements produced by MERGE, we want to say the opposite.
{X} = X
We want to identify singleton sets with their members

If MERGE involves an operation of removal or replacement (more on this below) such that a workspace containing *a* and *b*, notated [*a, b*] contains the set {*a, b*} with *a* and *b* removed from the workspace as the output of MERGE, then determining what is a head and what is not requires precisely the kind of cardinality-sensitive procedure sketched above. Sets are sets, and *a priori* there is no difference between a set with one member and a set with thirty unless you look inside the set and count its members. Let us make this more concrete. Suppose the workspace contains, as before, {*read*} and the set {*some, books*}:

14) Workspace: [{*read*}, {*some, books*}]

Then, MERGE replaces this pair of objects with a set containing them:

15) MERGE({*read*}, {*some, books*}) = [{{*read*}, {*some, books*}]

Determining whether {*read*} is a head involves (i) establishing its cardinality, and (ii) comparing that cardinality with the cardinality of the other set in the workspace, {*some, books*}. This is far from a trivial operation, and exactly how it could be accomplished is not clear in the current version of the theory. Collins (2017: 56) provides a picture that is compatible with our characterisation:

A ∈ {A, B} is final in SO iff there is no C contained in (or equal to) SO [Syntactic Objects] *such that A ∈ C, and C contains {A, B}. Otherwise, A is non-final in SO.*

Which is exactly what we have sketched above: it is necessary to be able to determine if a set is a singleton or not; the system must probe into a set, determine its cardinality to be greater than 1, and define it as 'final'/a 'head'.

If we do *not* want to identify a lexical item with a singleton (that is, if we go the opposite way with respect to the quotation from Chomsky, 2020a above), then when we have {read {some, books}} the system may simply find the object that is *not* a set: given Merge(A, B), establish which of those is not a set. If it is not a set, it is a head (i.e., a lexical item). There are multiple problems with identifying 'head' with 'lexical item' in the absence of a proper definition of 'lexical item' (e.g., do

we define them as terminal nodes? What do we do with idioms and other multi-word basic expressions?), but here we consider a purely formal issue: how exactly would MS determine that something is not a set? And, even if that could be solved (e.g., by some feature), would it not require a complication in the generative procedure? This last point can be fleshed out somewhat. Suppose that the only difference between XP and X is that the former is a set, but the latter is not (again, as the logical alternative to the scenario sketched in the previous paragraph). Then, Merge(H, XP) involves merging a non-set and a set; Merge(XP, YP) involves merging two sets; and Merge(H, H) involves merging two non-sets (the outputs of these operations are always sets, however). This means that Merge/MERGE should be formulated in such a way that it is specified that its output is always a set, but its input need not be: it can be a pair of non-sets, a pair of sets, or a set and a non-set. If Merge is limited to set + non-set situations, it would exclude any version of Pair-Merge (or equivalent operations to yield sister-adjunction and Chomsky-adjunction). More generally, it is not clear how we could introduce external arguments in the structure, since a DP subject would always be a set and the introduction of this DP would generate a set-set object: {DP, *v*P}. Interestingly, the configurations that have been dubbed 'problematic' (or 'ambiguous') for MS (e.g., in Chomsky, 2013, 2015; Epstein et al., 2015; Van Gelderen, 2019; Kitahara, 2020, among others) are the ones that involve either two sets ({XP, YP}) or two non-sets ({X, Y}). We will come back to this below, revisiting the notion of 'head' in graph-theoretic terms.

A further issue that impacts on the implementation of MS is that the identification of heads seems to be completely independent from semantics. Thus, in determining a label for the output of MERGE({*read*}, {*that*}), it is not sufficient to determine the cardinality of a set, since both sets are unary. Several options become available, most of which involve transforming a relation between two singletons into a relation between a singleton and a set with a greater cardinality: add more structure (possibly in the form of functional projections). There is no mention of the role of argument structure in the generative procedure in Chomsky (2019, 2020a, b), mainly because that would entail (as it did in the original Minimalist Program) that Merge can be motivated by the need to satisfy the valency of a predicate, against the idea that Merge is free, non-triggered (Chomsky, 2004 and subsequent works). The 1995 version of Merge was *asymmetric*:

> *The operation Merge(α , β) is asymmetric, projecting either α or β , the head of the object that projects becoming the label of the complex formed* (Chomsky, 1995: 227)

But, as we have seen, more recent developments have contested this asymmetry (MERGE, Simplest Merge, etc.). In this context, it is not clear how to interpret the claim that External Merge gives us argument structure, since argument structure is based on the idea of a predicate subcategorising for/selecting arguments. In other words, argument structure is inherently asymmetric: some things are

selected, some things are selectors. External Merge in the context of symmetric set formation may create configurations within which the valency of a predicate is satisfied, but that is a very different claim. In a further departure from theoretical developments in Chomskyan Minimalism, Minimalist Grammars (of the kind in Stabler, 2011 and related literature, see fn. 7) implement feature-driven Merge, which makes selectional requirements easy to model. The idea that Merge is simply unordered set formation brings about a number of complications, somewhat paradoxically.

The final difficulty that we will consider is precisely that search algorithms are defined for structured data. If the generative procedure is reduced to set formation, and those sets are unordered, then a search algorithm cannot apply: it would be akin to defining a search over a random array of data. If a search algorithm can be defined, it is because the data that constitutes the probing space for that algorithm is ordered. We have a first hurdle in a proper definition of MS: we need to specify the kind of formal objects defined by the generative operation (MERGE / Merge / Simplest Merge, etc.).

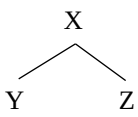
How can we apply a search mechanism in this scenario? One way is to impose an order to the sets, such that a search sequence can be defined. Considering only containment will not work: if we order syntactic objects in terms of proper containment, sister nodes cannot be ordered, since neither is a subset of the other. However, if the order in the output of Merge is given by labelling, such that {X, Y} is either {X, {X, Y}} or {Y, {X, Y}} (thus <X, Y> or <Y, X> by the pairing axiom; see e.g. Krivine, 1971: 2-3), then MS cannot be a pre-condition for a labelling algorithm, nor can MS be the labelling algorithm itself. If Merge/MERGE creates Chomsky-Collins sets, and a search algorithm is to be defined over those sets, they have to be ordered. That order comes with labels. However, if labelling is driven in some way by MS (Epstein et al., 2013, 2015, 2020; Chomsky, 2015; Vercauteren, 2017), then MS cannot be a search algorithm in the technical sense used in computer science: before labelling, the result of Merge is an unordered array.

We need to analyse labelling in some more detail, to the extent that it is perhaps the realm of the theory where MS features most prominently. Chomsky (2013, 2015) takes labelling to be the result of MS, which finds a head in a syntactic object formed by Merge. In this view, MS is part of the labelling algorithm. For a syntactic object SO,

> *Suppose SO = {H, XP}, H a head and XP not a head. Then LA will select H as the label, and the usual procedures of interpretation at the interfaces can proceed.* (Chomsky, 2013: 43)

Let us analyse this situation in some more detail. Suppose we have a local tree of the form

16)



Where Y and Z may be internally complex, i.e., Y may be the root of a sub-tree and so may Z. We will first explore the case where Y is a head and Z a non-head. Y is a symbol assigned to a category, which rules can operate over. So is Z. The question is, what is the indexed category assigned to an expression of the form [Y Z] (which will determine the kind of syntactic rules that can affect that object, the rules of semantic interpretation that will apply, its distributional properties, etc.)? That is the way in which we understand the problem. For example, if Y = *v* and Z = VP, then X = *v*P, because

> *minimal search finds v as the label of SO since v is unambiguously identifiable* (Epstein et al., 2015: 203)

Why cannot an object embedded in VP provide a label? Because

> *in any {H, XP}, the head H is always found with "less search" than any feature bearing (hence relevant information-bearing) element within XP.* (Op. Cit.)

This does not help in understanding how 'search' is implemented, but the idea sounds intuitive enough. It imposes some requirements over specific configurations: for example, an implementation of this idea requires us to assume that *carefully* in *read a paper carefully* cannot be introduced in the derivation as a head, but rather (i) as a full phrase or (ii) by means of some additional operation (e.g., Pair-Merge). This is so because if *read a paper* is a VP, Merge of (*carefully*, VP) would incorrectly label the resulting SO as AdvP. Chomsky (2020a: 48-49) proposes that MERGE comes in two flavours: symmetric and asymmetric:

> *In symmetrical MERGE, if you happen to have a head and an XP, then the head will provide the label – in earlier versions, what projects. But that's a case of MS (like most topics, not without some questions when we think about it carefully).*

This distinction seems to refer back to Set-Merge vs. Pair-Merge (Chomsky, 2004; see Langendoen, 2003 for a technical discussion), but if so, the new terminology is unexplained. It is unclear exactly what is *symmetrical* about {H, XP} constructions: in {*read*, {*some*, {*papers*}}, *read* subcategorises for an NP complement, not the other way around. From the point of view of linearisation, *read* asymmetrically c-commands *some papers*, and thus a procedure like Kayne's (1994) Linear Correspondence Axiom can apply yielding the string *read+some+papers* (where + is adjacency). How it is a case of MS is not explained. It is relevant to note that the original 1995 definition of Merge argues for the *asymmetry* of the operation:

> *The operation Merge(α , β) is asymmetric, projecting either α or β , the head of the object that projects becoming the label of the complex formed. If α projects, we can refer to it as the target of the operation*' (Chomsky, 1995: 225)

Labelling in the context of *bare phrase structure* (Chomsky 1994) was encoded in Merge itself, with some versions requiring Merge to be triggered by featural requirements (Di Sciullo & Isac, 2008; Wurmbrand, 2014; Pesetsky & Torrego, 2006; see also the references in fn. 6). Citko (2011) adopts the idea that labels are necessary in the syntax (thus rejecting explicitly the idea that Merge(X, Y) = {X, Y}), but allows for Merge(X, Y) to yield a 'complex label' {X, Y} for instances of Chomsky-adjunction: in the cases considered by Citko (2011: 178), X = Y. Despite being an argument in favour of symmetry in syntax, Citko-symmetry is clearly different from Chomsky-symmetry: there seems to be no symmetry in {H, XP} constructions in Citko's sense. But the precise nature of the relation between the symmetry or asymmetry of Merge and the definition of MS is unclear, not least because the role of MS in the kind of formal objects created by Merge is itself obscure.

In any case, the situation first explored by Chomsky (namely, {H, XP}) seems, *prima facie*, simple enough: if an object built by Merge contains a head and a non-head, the labelling algorithm, which works by MS, finds that head and labels the SO. However, formally expressing that is not a trivial task. The first question is how exactly the search would take place: in other words, how to define each step that makes up the 'algorithm'. This is not specified in the Minimalist literature. Above, when defining a search algorithm, we specified that it needs some way to compare inputs with the target of the search; we can adapt the simplest sequential search algorithm presented in Knuth (1998: 396) to the present context:

17) Given a sequence of syntactic objects SO_1, SO_2, …SO_n, the algorithm searches for a head.

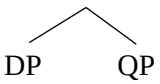
Step 1: Initialise. Set $i \leftarrow 1$
Step 2: Compare. If SO_1 is a head, terminate.
Step 3: Advance. Increase *i* by 1, proceed to S2.
Step 4: End of input – terminate.

It should be evident that the main problem with this algorithm (in addition to the issue, noted above, that if Merge yields unordered sets it is not possible to arrange terms in a unique sequence) is that the system must know somehow what a head is, and how to determine, given a syntactic object, whether it is a head or not. In set-theoretic syntax, as pointed out above, this requires at a minimum that the system be capable of evaluating the cardinality of a set. This is hardly something that 'blind' or 'free' Merge could do. Indeed, some computational implementations of the ideas in Chomsky (2013, 2015) need to do away with blind Merge (e.g., Ginsburg, 2016).

It is worth pointing out that in rewriting terms, the 'labelling' problem does not exist, because in a phrase marker like (16) X is introduced in the derivation one turn before Y and Z, and this introduction is precisely calling the indexed category to which X belongs. The problem of labelling

appears only if the derivation proceeds bottom-up, step-by-step (in other words: in a top-down model, labelling is not a problem). Here, it is necessary to provide a label to an object *after* that object has been created by Merge / MERGE, etc.; it has furthermore been proposed that syntactic objects may remain label-less (Chomsky, 1994; Collins, 2002; Citko, 2008; Hornstein, 2009). The idea of label-less objects pushes contemporary Minimalism even farther away from the well-defined grammars in canonical form that Chomsky himself studied in the '50s, and the advantages of pursuing such an approach are still unclear[7]. If a syntactic object is not labelled, they should not be able to be manipulated by syntactic operations, assuming that these are formulated as functions with an input and an output characterised as sequences of indexed categories. In set-theoretic terms, the 'label' is a member of the set, but in graph-theoretic terms a label is a node, the root of a subgraph. In other words: the *label* of a syntactic object is the node that is the root of the local tree that defines that syntactic object. Thus, if an operation takes that syntactic object as part of its input, the structural description of that operation will refer to the root of the syntactic object, not to every subpart of it. For example, if we have a rule of *NP movement*, the structural description of the rule does not refer to every possible set of nodes (and expressions in those nodes) that can be dominated (directly or transitive) by a node NP, it just refers to 'NP'. In set-theoretic terms, there has to be a way to refer to sets in a more abstract way; a *variable* over sets as it were. It is unclear how approaches with *unlabelled* objects actually make this work[8]. For example, Ott (2015: 194) provides the following derivation fragment: suppose that we have an object like (18) created by Merge:

18)

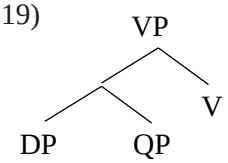


None of the objects is a head, so the LA cannot apply. The object in (18) gets (somehow) merged with V, yielding (19):

[7] It is worth noting that computationally explicit implementations of Minimalism, so-called Minimalist Grammars (Stabler, 2011; Graf, 2021; Michaelis, 2001; Gärtner & Michaelis, 2007; Kobele, 2009, among many others) assume a feature-rich system with a designated set of labels and projection indicated at every point in the tree (so does the Minimalist Machine in Fong & Ginsburg, 2020). There is, to the best of our knowledge, an implementation of MS-based labelling (and label-less Minimalism) in the context of Minimalist Grammars. Similarly, Ginsburg's (2016) proposal, inspired heavily in Chomsky (2013, 2015) finds itself forced to resign 'free Merge': *This type of free merge would create a huge computational burden because our model would have to compute an enormous number of unsuccessful derivations in order to arrive at a successful derivation. While free merge may have its merits (such as eliminating the need for movement operations to be motivated by feature checking, etc.), at this point, it is not clear to us how to implement free merge in a computationally efficient manner.* (fn. 14). Even when a root is 'unlabelled', it may enter further computations because the computational system is fed a single lexical item at a time; this presupposes the kind of Select operations that Chomsky et al. (2019: 245) consider unnecessary.

[8] In some recent Minimalist works (e.g., Chomsky, 2008; Ott, 2015), *label* is taken to be equivalent to *head*, which is a sense of *label* very different from the well-defined use in formal language theory (e.g., Hopcroft & Ullman, 1969: 19). In these works, an 'unlabelled' object is an object where no head can be 'detected by Minimal Search' (Ott, 2015). Without a formal definition of MS, this approach turns out to be hard to evaluate.

19)



At this point, we can ask exactly how the operation that produces (19) from (18) is formulated. How is the input for Merge specified? This is not clear, given that there is no obvious way to refer to the object in (18) as a unit for purposes of further computation (such as merge with V): the object in (18), as is, has no identifying address. What to do, then? Epstein et al. (2015), building on Chomsky (2013), propose that

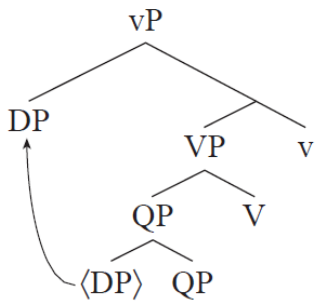
> *Suppose SO = {XP, YP}, neither a head. Here* ***minimal search is ambiguous****, locating the heads X, Y of XP, YP, respectively. There are, then, two ways in which SO can be labeled: (A) modify SO so that there is only one visible head, or (B) X and Y are identical in a relevant respect, providing the same label, which can be taken as the label of the SO.* (Epstein et al. 2015: 202. Our highlighting)

Collins (2017: 53) presents a very similar picture of Chomsky's LA:

> *If SO = {XP, YP} and neither is a head, then*
> *a. if XP is a lower copy, Label(SO) = Label(YP).*
> *b. if Label(XP) and Label(YP) share a feature F by Agree, Label(SO) = <F,F>.*

As in Moro (2000), a point of symmetry is broken via movement. In (20), then, after further structure is built, the DP may move, thus leaving the QP to be projected as a label:

20)



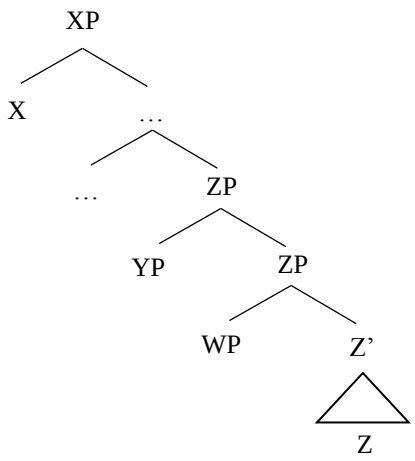
This process involves much backtracking: the object {DP, QP} must remain somehow active, requiring a label (despite the fact that it has entered further computations, which means there has to be a way to refer to it), and the system must be able to modify a previously generated object (this kind of operation required by the system in Chomsky, 2013, 2015, as noted in Ginsburg, 2016, is inherently counter-cyclic, with Ginsburg's proposal being '*an improvement over the manner in which feature checking occurs in POP*' by virtue of restricting probing to a root node, but not a true alternative to

counter-cyclicity). Among the multiple issues that emerge in this picture, we will solely focus on whether {XP, YP} and {X, Y} situations are indeed as problematic for a search algorithm as theoretical Minimalist work suggests.

2.1 What is 'minimal' in MS? Some considerations about 'distance' in Minimalism

How can we determine that a search has been 'minimal'? The grammar must be capable of determining the length of a path between nodes (and potentially comparing a number of paths to establish the shortest one). This is a weird thing to ask of the computational system (however it is formalised): once a suitable target has been found by some method, the only way to determine that this target is in some sense 'minimal' is to find a second target and compare the distances between the initial point of the search (possible, a probe) and target 1 and the same initial point and target 2. This seems a far cry from a 'perfect' faculty of language: why not stop the search on the first target?[9] Hornstein (2009: 38) proposes that *distance* is based on comparing the set of nodes involved in a movement path for possible targets, with shorter paths being properly contained in longer paths. The grammar must then contain a mechanism that evaluates the relation between two (or more) sets and is capable of identifying if one of those sets is a subset of the other (e.g., by defining an injective function from one into the other). Again, we are faced with the issue of requiring the grammar to be able to evaluate the cardinality of a set. This runs into difficulties, which can be illustrated with the use of *equidistance* in the formulation of operations like Agree. In these cases, there is a probe and multiple goals that, despite occupying distinct positions in a binary-branching phrase marker (and thus being more or less distant from the probe in terms of number of nodes/edges), are considered to be at the same distance from that probe for purposes of some specific operation. Let us illustrate the relevant configuration:

21)



[9] This issue arises more frequently than we may think. Consider economy principles like Minimal Link or Shortest Move: they are based on the assumption that short-distance dependencies are preferable to long-distance dependencies. However, unless a number of possible dependencies are compared in terms of length, it is unclear how exactly these principles (now subsumed to Third Factor/Minimal Computation principles) would actually work.

Let the *domain* of X as the set of nodes contained in the maximal projection of X (i.e., XP) which excludes X. Furthermore, let the *minimal domain* of Z be the set of categories that are only locally related to the head[10]. Consider now the case where X is a probe for movement: will it target YP or WP? Chomsky (1995: 169) says

> *If α, β are in the same minimal domain, they are equidistant from γ.*
>
> *In particular, two targets of movement are equidistant if they are in the same minimal domain.* (Chomsky, 1995: 169)
>
> (see also Lasnik, 2009; Hornstein 2009: 42, ff.; Boeckx, 2008: 145; for similar definitions of *equidistance*)

This means that YP and WP are equidistant for purposes of operations at X, despite the fact that, strictly speaking, YP c-commands WP asymmetrically. The theory of locality sketched in Chomsky (1995) requires this notion of *equidistance* for reasons related to feature-checking and movement (thus, intra-theoretical requirements). We may provide a couple of examples (which can also be found in the references above). In a case like

22) T seem [to him$_i$] [they$_k$ to like John$_{*i/j}$]

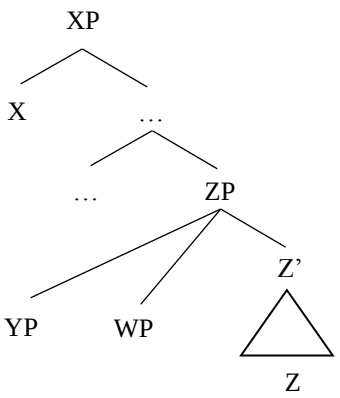
The question Chomsky considers is whether there is anything blocking raising of *they* to Spec-T: note that there seems to be a Principle C violation if *John* is coindexed with *him*, which suggests that *him* c-commands *John*; crucially, Chomsky assumes that *seem* '*has two internal arguments*' (1995: 280), the PP [to him] and the clause [they to like John] (i.e., the experiencer is not an adjunct). Under this assumption, a possible 'solution' (the details of which we will not consider here) is that *they* and *him* are equidistant, which allows *they* to move to Spec-T to check a Case feature without *him* being an intervening element in terms of Minimality.

If multiple specifiers of Z are equidistant to X since their paths are not proper subsets of each other (ZP need not be counted twice in each set, by the axiom of extensionality; see also Kitahara, 2020 for a different, but related, set-theoretic perspective), then (21) is equivalent to (23) (note that WP and YP are both neighbours of ZP):

[10] Symbolically,

> *The minimal domain Min(δ(CH)) of CH is the smallest subset K of δ(CH) such that for any γ ∈ δ(CH), some β ∈ K reflexively dominates γ.* (Chomsky, 1995: 274)

23)



(23) violates the venerable axiom of ‘binary branching all the way down’. This leads us to a further conundrum: extra structure between YP and WP is needed for linearisation purposes (if the process follows Kayne’s 1994 LCA; see also Uriagereka, 2012) but not for the computation of paths?

An interesting issue that emerges in the literature is that when MS needs to be made somewhat explicit, the issues we pointed towards in the previous section do appear, but the inconsistencies between maintaining strict set-theoretic unordered Merge and applying a search algorithm are not recognised. For instance, consider the following abstract structure and fragment, from Hayashi (2021: 22)

> *[(31)] {$_{\beta}$ X {$_{\alpha}$ Y {Z, W}}} (α=Y, β=X)*
>
> *[…] In (31), MS is required to refer to set β to locate the LI X: let us call it PATH (X) = β. Locating Y, in turn, requires MS to refer to sets β and α: PATH (Y) = (β, α). Here, PATH (X) is a proper subset of PATH (Y), where we can conclude that MS locates X prior to Y and that label β becomes X. In determining label α, X is irrelevant since it is not a member of set α. Since PATH (Y) = α and PATHs of the other competitors (Z and W, which are contained in the merge-mate set of Y) will include sets other than α, Y serves as label α.*

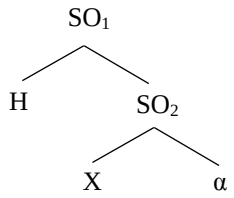
This approach is similar to Hornstein’s (2009), and as mentioned before, if understood in strict set-theoretic terms, requires the system to look into a set and determine the number of members. This is called a ‘path’: the ‘path’ of α is the set of all syntactic objects of which α is a term (thus, a path is a set of sets). In this context, and following Kitahara (2020), Hayashi (2021) characterises the ‘problematic’ objects for MS as in (24) (adapted from Hayashi, 2021: 22):

24) a. {$_{\alpha}$ X, Y} (both heads): Path(X) = Path(Y) = α
b. {$_{\gamma}$ {$_{\beta}$ X, …} {$_{\alpha}$ Y, …}}: Path(X) = (γ, β), Path(Y) = (γ, α)

In the cases in (24), neither ‘path’ is a subset of the other, thus the difficulties for MS. This way of calculating the success of MS is precisely the kind of procedure that is hard to picture unless the

grammar incorporates a counting and comparison operation. That is: in order for the syntactic component of the grammar to know that Path(A) is greater than, smaller than, or equal to, Path(B), for any A, B, it is necessary to count the number of sets that we need to refer to in order to locate a lexical item X within A, B. Kitahara (2020) explicitly recognises this: when considering a syntactic object like (25) (taken from Kitahara, 2020: 210)

25)

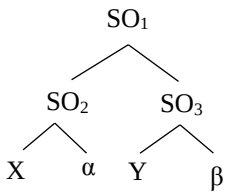


Kitahara claims that

> *MS selects H over X because the path of H (={SO₁}) is a proper subset of the path of X (={SO₁, SO₂}); hence, only H counts as an accessible head for labelling*

This is a curious claim. As noted before, establishing that Path(H) is a proper subset of Path(X) requires us to define *both*, and compare them. Why the computational component would define more than one search sequence after finding a suitable target is not clear. As we said before, the information that we have as external observers is not necessarily the one that the algorithm has access to. A system that includes these kinds of comparisons is akin to a generative grammar with transderivational constraints, like Minimal Link or Shortest Move (see Jacobson, 1997 for extensive discussion).
In addition to these issues, note that Kitahara's path comparison works only for structures generated via monotonic Merge: the structure must necessarily grow one terminal node at a time. Otherwise, paths cannot be defined as subsets of each other. For example, in (26) below (adapted minimally from Kitahara, 2020: 201) neither Path(X) and Path(Y) is a subset of the other:

26)



> Path(X) = {SO₁, SO₂}; Path(Y) = {SO₁, SO₃} (unordered set notation is used in Kitahara, 2020)

When this happens, Kitahara says (echoing Chomsky, 2013, 2015 and much related work), *both* objects are located by MS and '*count as accessible for labelling and valuation*'. This configuration is

the ‘problematic’ one for labelling, and these problems are claimed to be due to an ambiguity in MS. Is this really the case? We will explore this question in the following section.

3. Graphs and MS: the ‘ambiguity’ of {XP, YP} and {X, Y}

Above we introduced the concept of search algorithms for trees, in two variants: breadth-first and depth-first. We reviewed the difficulties of defining a search for unordered sets, but, what if Minimalist trees were taken as more than just graphical aids, notational variants of sets (diagrams of L-trees, in the words of Postal, 2010)? In other words, what if Minimalist trees were interpreted in all seriousness, as graphs? Assuming that MS is not an operation triggered by a head, the ‘search’ should start from the root. This is the path (pun intended) taken by Kitahara (2020): the Path(X) for any X is defined counting from the root. If there's a head involved in labelling (such that labelling a syntactic object is equated to finding the least embedded head within that syntactic object), then we are asking about the distance between that root R and a head H. However it is measured, let us notate that as *d*(R, H). If labelling is triggered by a specific head not much changes as far as the formalisation of searching goes, as we will see.

Then, we may characterise MS as a procedure that searches through a tree and finds a head H, and determines *d*(R, H): the length of the unique path that goes from R to H (see Krivochen, 2021 for extensive discussion about the properties of the workspace where such distance can be defined). In a multidominance digraph the walk is not unique, but it is still possible to define trails that define dependencies between expressions (binding, reconstruction, etc.); a proper definition of a search algorithm seems to be useful in the characterisation of phenomena that arise independently of set-theory based syntax.

If we follow the Minimalist proposals, once a head is found the algorithm terminates: in this sense, the qualification ‘minimal’ in MS seems superfluous (as observed in the previous section). This can be generalised for any specified target: once the algorithm finds an object in the input that matches its target, the search halts. If we go back to the definition of a path in graph-theory, as a sequence of nodes and edges in a digraph, Kitahara’s proposal can be readily formulated. As an additional advantage, in a rooted digraph there is always a unique path from R to any node. However, there is no need to establish a ‘preference’ between paths: if MS is indeed an algorithm, it halts as soon as it finds a key that coincides with the target. This brings up the additional problem of having the system know what it is looking for *a priori*, which we addressed above. Still, we have not provided a way for the algorithm to determine what a ‘head’ is: we did point out some problems that arise in a strict set-theoretical approach to syntax, but, can graph-theory make things easier?
In **Section 1.2** we introduced the notions of *indegree* and *outdegree* of a vertex, furthermore, we argued that since edges are part of the formal definition of a graph (as opposed to mere graphical aids

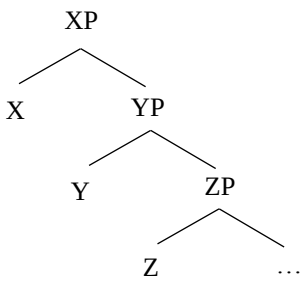
in the diagrammatic representation of a set), they have the same formal status as vertices. In this context, we can provide some revised definitions:

- The *root* of a tree is a node with indegree 0 and outdegree nonzero
- A *head* is a node with indegree non-zero and outdegree 0
- Intermediate nodes are nodes with indegree non-zero and outdegree nonzero

In the specific case of Minimalist structures, which are rigidly binary-branching and obey the so-called *Single Mother Condition*, then the outdegree of any node is either 0 or 2, and the indegree of a node is either 0 or 1[11]. We have now a graph-theoretic definition of *head* that uses nothing more than the tools already available to us to characterise the format of syntactic structures. MS, in this context, can be characterised as a search algorithm (possibly depth-first, possibly breadth-first) that, applying in a tree T (and assuming the SMC and binary branching just for the sake of exposition), defines a sequence Σ of alternating nodes and edges which specifies a unique walk from the root (the unique node with indegree 0 and outdegree 2) to a node with indegree 1 and outdegree 0: a 'head'.

Not much changes in terms of the straightforwardness of searching in monotonically growing trees: given a tree like (27)

27)



The search sequence from the root up until the first node with outdegree 0 would be Σ = <XP, X>, both under depth-first and breadth-first algorithms.

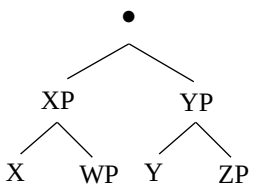
The first question that we want to address in this section is whether objects of the type {XP, YP} are indeed ambiguous for MS. Let us see how the search algorithms we saw before would work. Suppose that XP and YP both have a head and a complement, such that:

28) XP = {X, WP}
YP = {Y, ZP}

[11] This restriction is, however, not always empirically motivated: theories of multidominance and *n*-ary branching have been successfully used to describe linguistic phenomena and should not, in our opinion, be rejected on non-grammatical basis (e.g., reference to 'evolvability', 'perfection', etc.); see Lappin et al. (2000) for a related perspective.

And we merge XP and YP to create {XP, YP}, as in (29):

29)



An important aspect of (29) that we will come back to is that the p-marker has been construed assuming that we are dealing with a head-first language, where complements appear at the right of the heads that subcategorise for them. Given a structure like (29), suppose that the label of the node created by this merger, ●, is to be determined by MS. In this situation, a depth-first search over the structure just described would follow the sequence Σ until finding the first node with outdegree 0:

30) Σ = <●, XP, X>

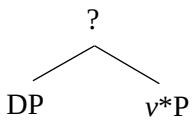
And a breadth-first search would follow Σ':

31) Σ' = <●, XP, YP, X>

Importantly, in neither case is there an ambiguity: when the algorithm finds an object that matches the target, the search stops. In both cases, depth-first and breadth-first, X is found, and the search should stop there. It is not necessary to count nodes, or to compare the distance between the root and X vs. the root and Y; if MS is only search, then no ambiguity arises. If the system can ignore a target and keep looking in a different branch of the graph, it is a departure from the simplest case that should (in accord with the Minimalist viewpoint) be justified. This may be the case if the specification of the target of the search includes the requirement that, in addition to being a node with outdegree 0, the target have a specific featural makeup. The same principle applies if the target of the search is something else: for instance, a node with indegree 1, outdegree 2, and a *wh*-feature. If an adequately restrictive meta-theory of features is devised (see Panagiotidis, 2021 for discussion), such a possibility is not unreasonable. What to do after that node has been found is a different matter: it is copied and re-merged at the root (i.e., movement)? Is a new edge created between the root and the target (i.e., multidominance)? Are some features copied and made into a label? The characterisation of MS in this paper is compatible with these and possibly other options: defining the search algorithm is independent from defining what the system does with what it has found. Note that in neither breadth-first nor depth-first searches is {XP, YP} problematic: if syntactic objects are defined in graph-theoretic terms, the search parameters can be characterised unambiguously. Importantly, it is not possible for a search algorithm (or for any computable function in a grammar in canonical form) to explore *both branches simultaneously*: from ● we cannot go at the same time to XP and to YP. That

would amount to simultaneity in rule application: this is a property of certain formal systems (e.g., cellular automata, Lindenmayer grammars) but not systems in canonical form; these include all formalisms in the Chomsky Hierarchy. In a tree like (30), it is true that both XP and YP are *accessible* for the search algorithm, but it does not mean that it is an ambiguous configuration that needs to be reorganised (e.g., via movement of one of the terms): either XP or YP will be probed into first. Mapping (29) into a structure where either XP or YP have moved is certainly possible; as a matter of fact, it seems to be the preferred way in the Minimalist literature. However, the justification for such movement is not found in the search algorithm, since there is nothing wrong with the configuration itself. The argument is equally valid if the search starts above ●. For example, if we take a concrete case of (29), adapted from Van Gelderen (2020):

32)

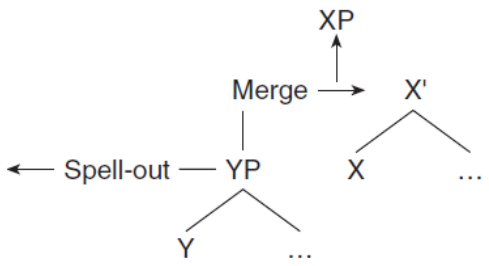


The theory in Chomsky (2013, 2015) and related works requires that (i) the new node created by merging DP and *v**P be left unlabelled, (ii) with the introduction of a subsequent head, T, DP moves to Spec-TP, leaving behind a copy, and (iii) the object in (32) is labelled *v**P because the copy would be 'invisible' for the labelling algorithm. It seems clear that this reasoning does not hold under the present view of how MS is defined, as a search algorithm: both the head D of DP and *v** of *v**P are 'accessible' (in the sense that they can be found by the algorithm), but one of them will necessarily be found first. If the search sequence is defined from left to right, then the head that will be found is D. If defined from right to left, it will be *v**. But moving DP in order to get a configuration where only *v** can be found is a stipulation that is independent from the formal definition of MS (plus the additional stipulation that a copies are invisible, which is an arcane point without a proper definition of 'visibility'; see also Van Gelderen, 2020 for an overview of criticisms of the theory in Chomsky, 2013, 2015).

Just as the definition of MS says nothing about what to do if a target is found (label, move, delete…), the question of exactly how much probing space is available for MS is independent from the definition of MS itself; rather, it depends on the system in which it is implemented. For example, MS in the context of Minimalism would be restricted by phase theory: only nodes within a tree with a phase head in its frontier would be within the search space of the algorithm. In this case, *v** and C (in the traditional version of phase theory) would delimit the set of accessible nodes. A radical version of this proposal would be Epstein & Seely's (2006: 12; Chapter 5) 'every phrase is a phase' (which would restrict MS to the extremely local level). ; Hegarty's (1993) proposal that functional heads also define elementary trees; traditional bounding nodes S and NP, etc. In other words: the definition of exactly how big the search space is is independent from the formal definition of MS.

Alternative visions of locality (which interests us to the extent that the theory of locality defines the probing space for MS) are of course possible, and may be preferable empirically. Let us give a couple of examples. Under strongly cyclic approaches, like Uriagereka's (2002, 2012) Multiple Spell-Out (MSO), or the lexicalised system explored in Krivochen (2019), the subject DP in a configuration like (32) would not be internally accessible for independent reasons. Because these systems are based on very different assumptions, it is useful to see how they restrict probing space in a way that is different from mainstream phase theory: both of these systems can be said to be interface-driven. In Uriagereka's system (in which Multiple Spell-Out is driven by linearisation), specifiers are Spelled-Out independently from the derivational cascade in which they are introduced, being assembled in a separate derivational space. Suppose that we have a phrase YP which needs to merge as the specifier of XP: a symmetry point between XP and YP does not allow an LCA-based linearisation algorithm to apply. Uriagereka's solution is to restrict the application of the LCA to local units where asymmetric c-command relations are unambiguous: these are called *command units*. We can diagram the order of operations in (33) (Uriagereka, 2002: 51):

33)



In Uriagereka's system, Spell-Out 'collapses' a phrase marker (here, YP) into something 'akin to a word' (Uriagereka, 2002: 49), without accessible internal syntactic complexity. This collapsed phrase marker can be merged to another syntactic term, once the linearisation paradox has been solved. MSO makes, essentially, a graph into a node (or, rather, a set of arbitrary cardinality into a singleton; Uriagereka, 2002: 50 explicitly takes set-theory seriously in his discussion). Specifiers and adjuncts are in distinct 'derivational cascades' with respect to the monotonically growing clausal spine of heads and complements; in the MSO framework, this makes them inaccessible for probing since probe and goal must be in the same derivational space.

In a lexicalised Tree Adjoining Grammar (LTAG; Frank, 2002; Joshi & Schabes, 1991; XTAG-group, 2001; Krivochen & García Fernández, 2019, 2020) the units of the syntactic computation are elementary trees defined as the extended projection of a single lexical head (the version in Krivochen & García Fernández specifies that only lexical *predicates* may define elementary trees); as pointed out before such elementary trees may be defined as the probing space for MS. As pointed out in Frank (2006, 2013), elementary trees need not coincide with phases: they may be bigger or smaller, depending on the presence of lexical heads. The lexicalised proposal explored in Krivochen (2019)

(building on the aforementioned works) restrict the size of the elementary units of the grammar to the extended projection of lexical predicates: the building blocks of syntax are graphs that contain a single lexical predicate, functional modifiers (e.g., temporal / aspectual auxiliaries), and arguments of that predicate. Elementary trees can be composed by means of generalised transformations: *substitution* (at the frontier or the root) and *adjunction* (at the non-root). Locality-wise, the proposal is that adjunction and substitution make a syntactic object equally opaque for operations triggered at the target of generalised transformations. Specifically,

> *Let γ and β be two elementary trees such that g contains a node X that corresponds to the root of β. A singulary transformation TS triggered from γ can affect β iff TS is intrinsically ordered after an embedding transformation that adjoins β to γ at X.* (Krivochen, 2019: 59)

We refer to this process as *atomisation* in Krivochen (2018a): a complex syntactic object becomes a computational atom. What singulary transformations cannot have access to is elements embedded within *β*; only *β* as a whole can be targeted by such operations. In Krivochen (2019) this provides a way of explanation for why resultative constructions are islands:

34) a. John hammered the metal flat / burnt the toast black
    b. The river froze solid
    c. Mary shouted herself hoarse
35) a. *What did John burn the toast / hammer the metal? (answer: 'black / flat')
    b. *What did the river freeze? (answer: 'solid')
    c. *What did Mary shout herself? (answer: 'hoarse')
36) a. */%…that / because, the metal, John hammered *t* flat.
    b. */%…that / because, the river, *t* froze solid.
    c. *…that / because, herself, Mary shouted *t* hoarse.

Summarising much discussion, the argument in Krivochen (2019) shows that if [the metal BECOME flat], [the river BECOME solid], and [Mary BECOME hoarse] are elementary trees derived in parallel with respect to the structures (technically, *initial trees*) [John hammered the metal], [the river froze], and [Mary shouted], then the adjunction of the former to the latter makes them opaque for extraction of syntactic terms embedded in the adjoined elementary trees, namely *flat*, *solid*, and *hoarse* (as exemplified in (35) and (36)). In the present context, it may be the case that generalised transformations (i.e., rules that relate distinct elementary trees) make auxiliary trees opaque for purposes of MS triggered at the initial tree that is targeted by adjunction. In both MSO and our cyclic lexicalised TAG approaches the crucial factor for the establishment of dependencies is that all terms of the relevant dependency (operator-variable, antecedent-anaphor, predicate-argument, etc.) be in the same elementary syntactic structure: a command unit in MSO (which derives the opacity of subjects

and adjuncts for purposes of extraction), an elementary tree in LTAG[12]. In graph-theoretic terms, MS may apply only within the boundaries of a local graph or be restricted to access the root of a distinct graph (but not embedded nodes).

The point is that whatever opacity conditions may hold for {XP, YP} cases, they can be defined independently of the definition of the search algorithm: these pertain to the probing space for the search, not to the formal definition of the search itself. Optimally, these constraints should have independent empirical support (subextraction from subjects in MSO, extraction from resultatives in our version of LTAG). Both MSO and adjunction-induced opacity ('atomisation') have been defined independently of search algorithms or labelling considerations, and on the basis of locality conditions that emerge in a strongly derivational minimalist framework (MSO) or a lexicalised graph-theoretic TAG (atomisation). MS is a tool that allows us to define dependencies, but these dependencies are independently constrained.

In any event, the important issue is the following: if a situation where two non-terminals are in a relation of sisterhood was really ambiguous for MS, and there was no external controller (in Turing's sense) the search procedure would just halt. But it doesn't, in formal characterisations of search algorithms. If MS doesn't just halt, then we need to assume (under standard computational assumptions) that there is either a bias in the algorithm that directs the branch that is to be looked at first or there is some external factor that dictates which branch is to be looked into first.

The same formal argument regarding MS is valid for {X, Y} situations: a node dominating two terminals (i.e., two nodes with indegree 1 and outdegree 0 each). However, some additional considerations must be made. A situation of the kind {X, Y} emerges, according to Chomsky (2013, 2015), in two cases:

i. When a root and a categoriser merge
ii. In instances of head movement

Both involve a number of intra-theoretical stipulations that do not really impact on the definition of MS (e.g., category-less roots, issues pertaining to head movement being post-syntactic, etc.).

[12] Frank (2002; 2013: 233) formulates this requirement explicitly:

> *Fundamental TAG hypothesis:*
> *Every syntactic dependency is expressed locally within a single elementary tree*

This also entails that all dependencies are local once recursive structure is factored out, something that a graph-theoretic approach can capture much better than a set-theoretic one.

Presumably, there could be other cases, depending on how much silent structure is there. For example, the bolded subsequences in (37a-b) could be said to involve two terminal nodes:

37) a. I **love her**
    b. John **reads it**

Under bare phrase structure assumptions, *love* and *her* should be two heads; merging them generates the object {*love, her*}. If this is correct, then monotransitive verbs with pronominal objects would generate {X, Y} situations, unless additional structure for the object is proposed in the form of functional nodes. For MS, there is no problem for the reasons explained above. In terms of labelling, a system where it is possible to say that the monotransitive verb selects an object, and the pronominal object partly satisfies the valency of the verb seems preferable to a system where the syntax is independent of such considerations (i.e., a theory of the grammar where it is possible to speak of relational networks is preferable to us, empirically, to an unconstrained generator endowed with arbitrary features). In this case, selection should play a role in labelling, if constituency is to be represented: *love her*, however it is encoded, behaves like an expression that can concatenate with an NP to yield a finite clause. In the categorial tradition, this is called an IV (intransitive verb phrase), phrase structure grammars traditionally use VP. The treatment of {X, Y} situations in the framework of Chomsky (2013, 2015) seems to require more theory-specific auxiliary hypotheses than {XP, YP}, and since they do not pertain to the definition of MS, we will not get into them in this paper.

4. Conclusions and some speculations

The aim of this paper was to provide a critique of the current Minimalist literature on MS and explore the possibilities to define MS as a search algorithm in a theory-neutral way. We argued that MS cannot be defined for set-theoretical structural descriptions, given current assumptions about free Merge and its unordered output. In other words, the strong set-theoretic commitments of mainstream Minimalism (which are not based on a comparison with formal alternatives for purposes of empirical analysis, but rather philosophical *a priori* conditions like evolvability and undefined 'economy') conspire against a definition of search algorithms over the output of Merge.

Search algorithms, and thus MS, can be defined over graphs: this entails a departure from the aforementioned set-theoretic commitments. The historically peripheral role that graph theory has played in the development of generative grammar with respect to operations over strings and sets (despite the early work on trees rather than strings and sets thereof as the model of structural descriptions; see e.g. Bach, 1964; McCawley, 1968; Zwicky & Isard, 1963) could be reversed as a result of pursuing lines of inquiry related to MS. In a graph-theoretic context, MS can be defined as either a breadth-first or a depth-first search. If we do, some assertions in the Minimalist literature with

respect to the 'problems' posed by {XP, YP} and {X, Y} objects for purposes of MS need to be revised[13].

The approach presented here must of course be refined. As part of our conclusion, we want to formulate some issues that set the agenda for future research, which at the moment amount to little more than speculation. The first issue that we want to address is if there is a way of determining whether MS is best modelled as a depth-first or a breadth-first algorithm. It is possible that deciding between depth-first or breadth-first is connected to other properties of the grammar. Consider, for instance, the approach to non-configurationality in Hale (1983), Austin & Bresnan (1996), and related works. If non-configurationality actually implies a radical reorganisation of phrase structure, such that syntactic representations in languages like Warlpiri or Jiwarli are 'flat' (as proposed in modular theories, whereby the representation of argumental relations is in charge of a morphological component, whereas the syntactic component can only generate *n*-ary branching phrase markers), a depth-first search may not be the best way to proceed: all targets may be at the same depth. In configurational languages, depth-first seems to be by and large assumed even in the informal characterisations of MS in the Minimalist literature. may be expressed in terms of preferring a breadth-first search, since structures are flat. In a system that is more rigidly organised, more 'configurational' in traditional parlance, depth-first seems to be more convenient. Choosing between depth-first and breadth-first may be a source of cross-linguistic variation.

As we mentioned above, an inherent problem in assuming something like MS for a set-theory based syntax is determining how exactly the system knows what a 'head' is. Either it is necessary to stipulate that (i) heads are not sets (and thus monotonic Merge manipulates a set and a non-set at every step) or (ii) the system can determine the cardinality of a set, with heads being singletons (i.e., sets of cardinality 1). Alternatively, 'headhood' can be encoded in feature form, but that is possibly the worst option given how arbitrary and non-explanatory it is. In graph-theoretic terms, however,

[13] There seems to be at least one {XP, YP} case that is somewhat exceptional: subject-predicate relations. Chomsky (2020a: 25) says that subject-predicate is an example of *exocentric* construction (see also Chomsky, 2013), which would yield a structure very much in line with early generative grammar's S → NP, VP (or NP, PredP in slightly later developments). The case of exocentric structures poses yet another problem for MS-driven labelling: how is it possible to label something by MS if the intended label is not a 'projection' (or percolation, or copy…) of features of any of the syntactic objects involved in the search? In other words: what is the target for MS? If subject-predicate objects are not labelled by MS, does that mean that there is a different kind of labelling? Or, if subject-predicate objects are not labelled at all, how can they enter further computations? Suppose that this issue is circumvented by moving the subject to a higher position (say, Spec-TP), thus leaving a trace or a copy, and labelling as whatever remains. In order to have this movement, the head T must merge with the symmetric object {DP, *v*P}, which is unlabelled. At this point, {DP, *v*P} must be treated as a unit for purposes of Merge in order to get the operation to remain binary: we can get that if Merge creates rooted trees, such that every application of Merge results in an object with two edges and three vertices but not if syntactic computation is set-formation. The root of that local tree may not be assigned a category, but it is a node in the graph and it bears an identifier, such that it can be used as part of the input for further processes. However, this object also must be internally complex, such that the DP is accessible and (under traditional Minimalist assumptions) can be extracted and re-merged at Spec-TP.

determining headhood is a matter of establishing if a node has outgoing edges. Note that edges are part of the formal definition of a graph: they are not merely parts of diagrams. Thus, if the system builds graphs, edges are part of the set of objects it operates over. A 'head' in this context is a node in a digraph that has no outgoing edges, the root of the digraph is a node that has no incoming edges, and intermediate nodes have both outgoing and incoming edges. Whether intermediate nodes (which correspond to derived but non-final objects) are required for syntactic representations is, again, orthogonal to the definition of MS. Models like Arc Pair Grammar and Metagraph grammar (Postal, 2010), for example, do not use intermediate nodes in the sense PSGs (and some forms of Minimalism) do; this does not interfere with the possibility of defining search operations over representations in those models.

In the context of defining MS, an issue that emerged is that given a bifurcation, the system must decide which branch to explore first. One of the branches will be explored first, and there is no ambiguity in the definition of the algorithm, but the issue can be turned into an empirical question given specific assumptions. In the formal work on search algorithms, given a *preorder* transversal through a tree, a sequence *root-left-right* is assumed; however, linguistic considerations may allow for a *root-right-left* transversal under certain conditions. This is an issue that is independent of the formal definition of MS, but worth exploring in terms of the linguistic applications of MS. Under more or less standard Minimalist assumptions, if MS is looking for heads, the bias may be implemented in a parametric system as a so-called *head parameter*: in traditional PS trees, head-initial structures branch to the right, whereas head-final structures branch to the left. This can be expressed in the following terms:

- Head-initial language: given a symmetry point, MS is biased to explore the leftmost branch first
- Head-final language: given a symmetry point, MS is biased to explore the rightmost branch first

This is an oversimplification, however. While there are languages that are consistently head-initial or consistently head-final (e.g., Turkish or Japanese are usually characterised as consistently head-final), most languages show both head initial phrases and head final phrases (e.g., English is head-final in NPs, and head-initial in VPs). The level of granularity at which branch preference should be defined in order to implement this idea seems to be construction-specific.

However it is modelled, the decision of what branch to explore first is not part of the formal definition of MS, but part of an appropriate characterisation of its implementation.

The definition of MS in this paper aims to contribute to the debate of the proper format of syntactic representations, in particular siding with those who follow a graph-theoretic approach: issues of MS

are not necessarily exclusive to Minimalist syntax. We suggest that if operations in Minimalism are as based on MS as the literature suggests, then it may be both theoretically and empirically fruitful for Minimalists to take graphs seriously.

5. References

Austin, Peter and Joan Bresnan (1996) Non-configurationality in Australian aboriginal languages. *Natural Language and Linguistic Theory* 14. 215–268.

Bach, Emmon (1964) *An Introduction to Transformational Grammars*. New York: Holt, Rinehart & Winston.

Boeckx, Cedric (2008) *Aspects of the syntax of agreement.* London: Routledge.

Chomsky, Noam (1994) Bare phrase structure. *MIT occasional papers in linguistics* 5.

Chomsky, Noam (1995) *The Minimalist Program*. Cambridge, Mass.: MIT Press. [cited by the 2nd Edition, 2015]

Chomsky, Noam (2004) Beyond Explanatory Adequacy. In Adriana Belletti (ed.) *Structures and Beyond.* Oxford: OUP. 104–131.

Chomsky, Noam (2013) Problems of Projection. *Lingua* 130. 33-49.

Chomsky, Noam (2015) Problems of Projection: Extensions. In Elisa Di Domenico, Cornelia Hamann & Simona Matteini (eds.) *Structures, Strategies and Beyond: Studies in honour of Adriana Belletti*. Amsterdam: John Benjamins. 1-16.

Chomsky, Noam (2019) Some Puzzling Foundational Issues: The Reading Program. *Catalan Journal of Linguistics*. 263-285.

Chomsky, Noam (2020a) The UCLA lectures. https://ling.auf.net/lingbuzz/005485

Chomsky, Noam (2020b) Minimalism: where we are now, and where we are going. Lecture at 161st meeting of Linguistic Society of Japan. Available online: https://www.youtube.com/watch?v=X4F9NSVVVuw

Chomsky, Noam, Angel Gallego & Dennis Ott (2019) Generative Grammar and the Faculty of Language: Insights, Questions, and Challenges. *Catalan Journal of Linguistics*. 229-261.

Citko, Barbara (2008) Missing labels. *Lingua* 118. 907–944.

Citko, Barbara (2011) *Symmetry in syntax: Merge, move, and labels*. Cambridge: CUP.

Collins, Chris & Edward Stabler (2016) A Formalization of Minimalist Syntax. *Syntax*, 19(1). 43-78.

Collins, Chris (2002) Eliminating Labels. In Samuel D. Epstein and T. Daniel Seely (eds.), *Derivation and Explanation in the Minimalist Program*. Oxford: Blackwell. 42-64.

Collins, Chris (2017) Merge(X, Y) = {X, Y}. In Leah Bauke & Andreas Blühmel (eds.) *Labels and Roots*. Berlin: Walter de Gruyter. 47-68.

Collins, Chris & Erich Groat (2018) Distinguishing Copies and Repetitions. http://ling.auf.net/lingbuzz/003809

Cormen, Thomas, Charles Leiserson, Ronald Rivest, & Clifford Stein (2001) *Introduction to Algorithms*. [2nd Edition]. Cambridge, Mass.: MIT Press.

Di Sciullo, Anna-Maria & Daniela Isac (2008) The Asymmetry of Merge. *Biolinguistics* 2.4: 260–290.

Epstein, Samuel D. & T. Daniel Seely (2006) *Derivations in Minimalism*. Cambridge: CUP.

Epstein, Samuel; Husatsugu Kitahara & T. Daniel Seely (2013) Structure building that can't be! In Myriam Uribe-Etxebarria and Vidal Valmala (eds.) *Ways of structure building*. Oxford: OUP.

Epstein, Samuel; Husatsugu Kitahara & T. Daniel Seely (2015) Labeling by Minimal Search: Implications for Successive-Cyclic A-Movement and the Conception of the Postulate "Phase". In *Explorations in Maximizing Syntactic Minimization*. London: Routledge. 201-221.

Epstein, Samuel; Hisatsugu Kitahara & T. Daniel Seely (2020) Unifying Labeling under Minimal Search in "Single-" and "Multiple-Specifier" Configurations. *Coyote Papers: Working Papers in Linguistics*, 22. 1-11.

Even, Shimon & Guy Even (2012) *Graph algorithms* [2nd Edition]. Cambridge: CUP.

Fong, Sandiway & Jason Ginsburg (2020) *Minimalist machine*. http://elmo.sbs.arizona.edu/sandiway/mpp/mm.html

Frank, Robert (2002) *Phrase Structure Composition and Syntactic Dependencies*. Cambridge, Mass.: MIT Press.

Frank, Robert (2006) Phase theory and Tree Adjoining Grammar. *Lingua*, 116(2). 145–202.

Frank, Robert (2013) Tree adjoining grammar. In Marcel den Dikken (ed.) *The Cambridge Handbook of Generative Syntax*. Cambridge: CUP. 226-261.

Gartner, Hans-Martin & Jens Michaelis (2009) Locality Conditions and the Complexity of Minimalist Grammars: A Preliminary Survey. *Model-Theoretic Syntax at 10, Proceedings of the ESSLLI-Workshop*. 87-98.

Graf, Thomas (2021) The computational unity of Merge and Move. To appear in *Evolutionary Linguistic Theory*.

Ginsburg, Jason (2016) Modeling of problems of projection: A non-countercyclic approach. *Glossa* 1(1): 7. 1-46. http://dx.doi.org/10.5334/gjgl.22

Hale, Kenneth (1983) Warlpiri and the grammar of non-configurational languages. Natural Language and Linguistic Theory 1, 5-47.

Hayashi, Norimasa (2021) *Labels at the Interfaces: On the Notions and the Consequences of Merge and Contain*. PhD dissertation, Kyushu University.

Hegarty, Michael (1993) *Deriving clausal structure in tree adjoining grammar*. Unpublished ms., University of Pennsylvania.

Hornstein, Norbert (2009) *A theory of syntax: minimal operations and Universal Grammar*. Cambridge: CUP.

Jacobson, Pauline (1997) Where (if anywhere) is transderivationality located? In Peter Culicover and Louise McNally (eds.) *The Limits of Syntax*. New York: Academic Press. 303-336.

Joshi, Aravind K. & Yves Schabes (1991) Tree-Adjoining Grammars and Lexicalized Grammars. *Technical Reports (CIS).* Paper 445. http://repository.upenn.edu/cis_reports/445

Kahane, Sylvain & François Lareau (2016) Word ordering as a graph-rewriting process. In Annie Foret, Glyn Morrill, Reinhard Muskens, Rainer Osswald & Sylvain Pogodalla (eds.) *Formal Grammar*. Berlin: Springer. 216–239.

Kayne, Richard (1994) *The antisymmetry of syntax*. Cambridge, Mass.: MIT Press.

Kitahara, Hisatsugu (2020) 'Multiple specifier' configurations revisited. *Reports of the Keio Institute for Cultural and Linguistic Studies* 51. 207-216.

Knuth, Donald (1998) *The art of computer programming Vol. 3: Sorting and Searching*. [2nd Edition]. Reading, Mass.: Addison-Wesley.

Kobele, Gregory (2009) Syntax and semantics in minimalist grammars. ESSLI '09. https://esslli2009.labri.fr/documents/Kobele06-2.pdf

Komachi, Masayuki, Hisatsugu Kitahara, Asako Uchibori, and Kensuke Takita (2019) Generative procedure revisited. *Reports of the Keio Institute of Cultural and Linguistic Studies* 50 (2019), 269-283.

Krivine, Jean-Louis (1971) *Introduction to axiomatic set theory*. Dordrecht: Reidel.

Krivochen, Diego Gabriel (2015) Copies and Tokens: Displacement Revisited. *Studia Linguistica* 70(3). 250-296. https://doi.org/10.1111/stul.12044

Krivochen, Diego Gabriel (2018a) *Aspects of emergent cyclicity in language and computation*. PhD dissertation, University of Reading.

Krivochen, Diego Gabriel (2018b) *Syntax on the edge*: *a graph-theoretic analysis of English (and Spanish) sentence structure*. Ms. https://ling.auf.net/lingbuzz/003842

Krivochen, Diego Gabriel (2019) On trans-derivational operations: Generative Semantics and Tree Adjoining Grammar. *Language Sciences* 74. 47-76. https://doi.org/10.1016/j.langsci.2019.04.002

Krivochen, Diego Gabriel (2021) Towards a theory of syntactic workspaces: neighbourhoods and distances in a lexicalised grammar. Ms. Under review.

Krivochen, Diego Gabriel & Luis García Fernández (2019) On the position of subjects in Spanish periphrases: Subjecthood left and right. *Borealis: An international journal of Hispanic linguistics*, *8*(1), 1-33. https://doi.org/10.7557/1.8.1.4687

Krivochen, Diego Gabriel & Luis García Fernández (2020) Variability in syntactic-semantic cycles: evidence from auxiliary chains. In Melvin González-Rivera & Sandro Sessarego (eds.) *Interface-Driven Phenomena in Spanish: Essays in honor of Javier Gutiérrez-Rexach*. London: Routledge. 145-168.

Lakatos, Imre (1978) *The Methodology of Scientific Research Programmes* [Vol. 1]. Cambridge: CUP.

Langendoen, D. Terence (2003) Merge. In *Formal Approaches to Function in Grammar: In Honor of Eloise Jelinek*, ed. by Andrew Carnie, Mary Willie and Heidi Harley, 307-318. Amsterdam: John Benjamins.

Lappin, Shalom; Robert D. Levine & David Johnson (2000) The Structure of Unscientific Revolutions. *Natural Language & Linguistic Theory* 18(3). 665–671.

Larson, Bradley (2015) Minimal search as a restriction on Merge. *Lingua* 156. 57-69.

Lasnik, Howard (2009) Shortest Move and Equidistance. Linguistics 611 class notes, University of Maryland.

McCawley, James (1968) Concerning the base component of a transformational grammar. *Foundations of Language* 4. 243-269.

McKinney-Bock, Katherine & Jean-Roger Vergnaud (2013) Grafts and beyond: Graph-theoretic syntax. In Katherine McKinney-Bock & María Luisa Zubizarreta (eds.) *Primitive Elements of Grammatical Theory*. London: Routledge. 207-236.

Michaelis, Jens (2001) Derivational Minimalism is Mildly Context-Sensitive. In M. Moortgat (ed.), *Logical Aspects of Computational Linguistics (LACL '98), Lecture Notes in Artificial Intelligence* Vol. 2014. 179-198.

Moro, Andrea (2000) *Dynamic antisymmetry*. Cambridge, Mass.: MIT Press.

Ott, Dennis (2015) Symmetric Merge and Local Instability: Evidence from Split Topics. *Syntax* 18(2). 269-303.

Panagiotidis, Phoevos (2021) Towards a (minimalist) Theory of Features: preliminary notes. Ms. https://ling.auf.net/lingbuzz/005615

Pesetsky, David and Torrego, Esther (2006) Probes, goals and syntactic categories. In *Proceedings of the 7th Annual Tokyo Conference on Psycholinguistics*. Tokyo: Hituzi Syobo Publishing Company. 25–60.

Postal, Paul M. (1998) *Three Investigations on Extraction*. Cambridge, Mass.: MIT Press.

Postal, Paul M. (2004) *Skeptical Linguistic Essays*. Oxford: OUP.

Postal, Paul M. (2010) *Edge-Based Clausal Syntax*. Cambridge, Mass.: MIT Press.

Sedgewick, Robert & Kevin Wayne (2014) *Algorithms* [2 vols.]. NY: Addison-Wesley.

Stabler, Edward (2011) Computational perspectives on minimalism. In Cedric Boeckx (ed.) *Oxford handbook of minimalism*. Oxford: Oxford University Press. 617–641.

Stephen, Graham (1994) *String searching algorithms*. London: World Scientific.

Uriagereka, Juan (2002) Multiple Spell-Out. In Uriagereka, J. *Derivations: Exploring the Dynamics of Syntax*. London: Routledge. 45-65.

Uriagereka, Juan (2012) *Spell-Out and the Minimalist Program*. Oxford: OUP.

Van Gelderen, Elly (2019) Cyclical Change and Problems of Projection. In Anne Breitbarth, Miriam Bouzouita, Lieven Danckaert, and Elisabeth Witzenhausen (eds.) *Cycles in Language Change*. Oxford: OUP. 13-32.

Van Gelderen, Elly (2020) *Third factors in syntactic variation and change*. To appear in CUP.

Van Steen, Maarten (2010) *Graph Theory and Complex Networks: An Introduction*. Available for free at https://www.distributed-systems.net/index.php/books/gtcn/gtcn/

Vercauteren, Aleksandra (2017) Features and labeling: Label-driven movement. In Leah Bauke & Andreas Blühmel (eds.) *Labels and Roots*. Berlin: Walter de Gruyter. 69-90.

Wilson, Robin (1996) *Introduction to Graph Theory*. [4th edition]. London: Adison Wesley.

Wurmbrand, Susi (2014) The Merge Condition: A syntactic approach to selection. In Peter Kosta, Steven L. Franks, Teodora Radeva-Bork and Lilia Schürcks (eds.) *Minimalism and Beyond: Radicalizing the interfaces*. Amsterdam: John Benjamins. 130-167.

XTAG group (2001) A lexicalized TAG for English. Technical report, University of Pennsylvania. https://repository.upenn.edu/cgi/viewcontent.cgi?article=1020&context=ircs_reports

Zwicky, Arnold & Stephen Isard (1963) Some aspects of tree theory. Working Paper W-6674, The MITRE Corporation, Bedford, Mass. Available at: https://web.stanford.edu/~zwicky/some-aspects-of-tree-theory.pdf