Towards a theory of syntactic workspaces: neighbourhoods and distances in a lexicalised grammar

Diego Gabriel Krivochen

diegokrivochen@hotmail.com

Abstract:

Syntactic operations in generative grammar often require to keep syntactic objects active for a number of derivational steps (Copy, Agree, Move); however, only recently has the importance of formalising *where* those syntactic objects are kept active received some attention. Contemporary generative theory (e.g., Chomsky, 2019, 2020) has somewhat acknowledged that syntactic operations must apply *somewhere*, particularly when *copying* and *movement* operations are considered: recent work on structure building and mapping (MERGE) makes explicit reference to *workspaces*. However, it is still a vastly underexplored area. In this paper we explore the concept of *workspace* and its role in current generative theory, aiming at a precise characterisation of what workspaces are and how their properties determine possible syntactic configurations using tools from topology and graph theory. We analyse the consequences of conceptualising 'syntax' as a set of operations that affect local regions of spaces.

Keywords: Syntax; workspace; phrase markers; copy; topological space

1. Introduction

In transformational generative grammar, a recurrent topic has been the need to *hold on* to structure, either because it needs to be kept within probing memory for further operations (for instance, indexing) or because it has been subject to a reordering rule. We can exemplify these cases in (1a-b)

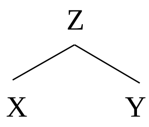
1) a. Mary$_i$ thinks [that Peter likes [an old picture of herself$_i$]]

b. Which picture of herself does Mary think that Peter likes ~~which picture of herself~~?

The non-local dependencies in (1), one base-generated co-indexing and the other a product of movement, illustrate the necessity of keeping structure active for a number of derivational steps. But if we look closely at this process, things are unclear. Operations that make reference to previous derivational steps (i.e., not to the last line of rewriting, or the root node of a binary-branching tree), or to chunks of structure, require syntactic objects of variable complexity to be *stored* somewhere, where they can be accessed and retrieved. In connection to this, recent generative theory has appealed to the idea of *workspaces*, and they are playing an increasingly important role in the definition of syntactic operations and dependencies (e.g., Müller, 2004: 298; Kato et al., 2016; Collins & Stabler, 2016; Jayaseelan, 2017; Chomsky, 2019, 2020; Chomsky et al., 2019; Komachi et al., 2019; Epstein et al., 2020), but very often without deep formal discussion or definitions: a detailed discussion about what a workspace actually is, what adding a workspace to the grammar actually means, and what the consequences for the architecture of the grammar and the generative power of the system are is still missing in contemporary generative theory. The purpose of this paper is to make explicit exactly what thinking about *workspaces* commits us to in formal terms, and how we can make use of mathematically explicit characterisations of spaces to our advantage in syntactic theory. In order to do this, we will need to define what spaces are, which in turn will require the introduction of some core notions in topology. This is important and necessary since the tools that topology puts at our disposal can help us define the properties of spaces that are relevant for syntactic computations and therefore evaluate recent proposals critically at both theoretical and empirical levels.

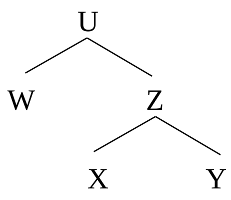
It is important to note that viewing phrase markers as topological objects is not necessarily a new idea: already Bach (1964: 71) formulates conditions on phrase markers (P-markers) defined as '*topological structure*[s] *of lines and nodes*'. This perspective allowed for the formalisation of conditions over structural descriptions in graph-theoretical and geometrical terms (e.g., Zwicky & Isard, 1967; McCawley, 1968; Morin & O'Miley, 1969; Kuroda, 1976; Arc Pair Grammar; Johnson & Postal, 1980 and its spiritual successor, Metagraph Grammar; Postal, 2010; see Kracht, 2001; Beim Graben & Gerth, 2012 for a more mathematical perspective). In these works, operations apply to nodes, creating or deleting edges, in order to establish syntactic dependencies. The graph-theoretic inclinations of early generative grammar contrasts with the Minimalist custom of defining syntactic objects in set-theoretic terms (Chomsky, 2008, 2019; Epstein et al., 2015; Collins, 2017); some objects that can be defined in graph-theoretic terms become impossible to define in set-theoretic terms, and this will become relevant below. Let us see a simple example from a transformational perspective. Assume that we have a phrase marker in which objects X and Y, introduced in the derivation by discrete recursive combinatorics, are in a local relation, as represented in (2):

2)



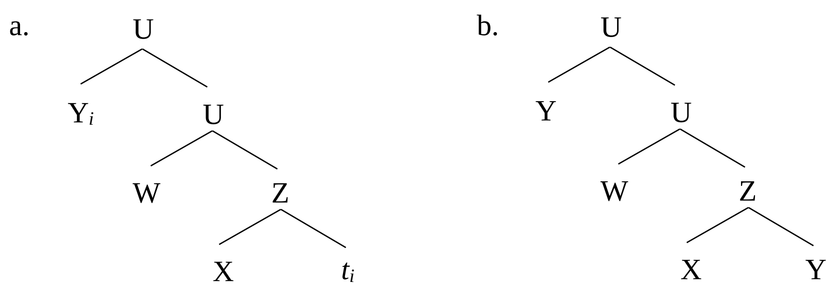
Now suppose that there is some relation *R* between X and Y: for instance, say X theta-marks Y. That relation needs to be maintained throughout the derivation, or reconstructed at the level of semantic interpretation if disrupted by a reordering or deletion rule. Let us now introduce a further element in the derivation, W, which requires a local relation with Y in order to satisfy some requirement (which one in particular is not relevant for the present argument). W is external to {X, Y}, following a cumulative approach to derivational dynamics (which Chomsky, 1995: 190 encodes in the so-called *Extension Condition*):

3)



But, what happens if a local configuration between W and Y is required (because, for instance, Y satisfies a feature on W), and such relation cannot hold if X is in between? A *displacement-as-movement* approach can either (a) move Y to a higher position in the checking domain of W (extending U), outside the scope of X leaving a co-indexed trace behind (the so-called *trace theory*), or (b) *copy* Y and *re-introduce* Y in the derivation (the so-called *Copy Theory of Movement* CTM, or *Copy+Re-Merge theory*; Chomsky, 2000; Uriagereka, 2002; Nunes, 2004 and much related work). Both options are diagrammed below:

4)



In both cases, the structure is extended by means of extra nodes: in (4a), we add a *trace* of Y, an index to Y and *t*, and expand U; in (4b) we add a copy of Y and similarly expand U. In both cases, there is a local relation between W and Y, as required (because there is no other head between these two nodes), but at the cost of introducing extra inaudible structure (non-terminal nodes). Moreover, the very idea of *copying* requires not only an operation that takes Y and somehow produces another Y, but this has to happen *somewhere* (where the copy is stored and where it can be retrieved from): this *somewhere* has usually been referred to as a *workspace* an explicit workout of the concept of *workspace* seems to be unavoidable, yet there is no explicit characterisation of what *workspaces* are and how they interact with the generative procedure in mainstream Minimalism (Mainstream Generative Grammar, or MGG, henceforth). To give a recent example, Chomsky et al. (2019: 236, 245) explicitly say that

MERGE operates over syntactic objects placed in a workspace: the MERGE-mates X and Y are either taken from the lexicon or were assembled previously within the same workspace

All syntactic objects in the lexicon and in the workspace WS are accessible to MERGE; there is no need for a SELECT operation (as in, e.g., Chomsky 1995). WS represents the stage of the derivation at any given point.

More formally, Chomsky defines MERGE as follows (personal communication cited in Komachi et al., 2019: 275; see also Chomsky, 2020: 34, 42):

MERGE(P, Q, WS) = [{P, Q}, X_1, ..., X_n] = WS', where if Y is a term of WS [WorksSpace], *it is a term of WS'.*

In this context, the lack of specific accounts of the properties of the workspace where operations are supposed to apply, or even an explicit definition of *what a workspace is* is rather surprising, and many questions arise: does the stage of the derivation at any given point include all objects in the lexicon[1]? How is lexical insertion formulated in such a system (among other issues: without any procedure to select a lexical array or numeration, how can the grammar know when to generate *John loves Mary* and when *Peter broke the vase*)? Chomsky et al. (2019) do not specify how a derivation would begin: in the definition of MERGE(P, Q, WS) (e.g., Chomsky, 2020: 48; Komachi et al., 2019: 275), how are P and Q selected? The only thing that is specified is that P and Q are accessible in the workspace, which does not answer this particular question. What prevents the system from overgenerating or having to search through the whole lexicon every time a new terminal node is to be added to a phrase marker? What are the consequences of this assumption for algorithms of probing within the workspace? (e.g., the establishment of filler-gap dependencies), among others. These questions cannot be answered if we do not first know exactly what a workspace is. The absence of detailed discussion about the nature and properties of the workspace is also concerning because the properties of the workspace may impose conditions on the operations that can apply; if syntactic operations apply to objects in a space, those objects need to be characterised as arrays of components in that space. In turn this has deep consequences for an account of *dependencies* between objects: the very notions of *local* and *non-local* dependency need to be reconceptualised depending on what is available at every derivational step. For definitions, Chomsky et al. (2019: 236) refer the reader to Collins & Stabler (2016: 46), who say:

> ***Definition 10.*** *A stage (of a derivation) is a pair S = <LA, W>, where LA is a lexical array and W is a set of syntactic objects. In any such stage S, we will call W the workspace of S.* […] *by*

[1] The answer to this question is *yes*, e.g. in Kato et al. (2016: 35): *We assume that WS is the set consisting of SOs already constructed* ***and LIs in the Lexicon****, that is, WS = {$\Sigma_1, \ldots, \Sigma_n$} ∪ Lexicon = {$\Sigma_1, \ldots, \Sigma_n$, $LI_1, \ldots, LI_m$}.* Also in Chomsky (2020: 45): *for a given I-language, the set of workspaces¬the set notice, not the least set¬is the set containing* ***the lexicon*** *and containing MERGE (P, Q, WS) for any P, Q and WS that has already been generated* (our highlighting).

convention we will reserve the term "syntactic object" for those elements built up in the course of the derivation and contained in the workspace.

In this context we can consider Chomsky's view in comparison to Collins & Stabler's, as they represent what can be found elsewhere in the literature (e.g., Müller's 2004: 298 goes along the lines of Collins & Stabler's[2]). It is unclear whether the workspace is, to give just a couple of examples, (i) a syntactic object (as in Collins & Stabler, 2016: 47; or a set thereof, see Komachi et al., 2019, who work with workspaces defined along these lines, as per Chomsky's definition of MERGE cited above), (ii) a set including not only syntactic objects but also other workspaces or sets (e.g., the lexicon, as a set of lexical items or features), or (iii) a buffer / memory stack / working memory where syntactic operations apply or where elements are stored and accessed[3]. Part of this confusion arises from the lack of explicitness about whether a set of syntactic objects is always a syntactic object. Collins & Stabler (2016: Definition 7) explicitly say it is, and therefore a workspace is a syntactic object (Op. Cit.: 47). It is not clear, however, whether in Chomsky's (2019, 2020) version of the theory a set of syntactic objects is a syntactic object (in Chomsky, 1995 every syntactic object is either a lexical item or a labelled set of lexical items, but it is not clear whether any set of syntactic objects is a syntactic object). These issues are fundamental, but remain unaddressed in the literature. Several questions arise. If the workspace of a derivation is the structure already built (or the structure plus the *entirety of the lexicon*, as in Kato et al., 2016: 35; Chomsky, 2020), then how does it help at

[2] Specifically, Müller (2004: 298) says:

> *The workspace of a derivation D comprises the numeration N and material in trees that have been created earlier (with material from N) and have not yet been used in D.*

Note that in this case, the grammar must include a procedure to probe into the generated structure, compare it with the numeration, and establish which elements have or have not been used yet. Or, include numerical indices in the numeration indicating how many times each item is to be used (which should still be accessible for a computational procedure at every step of the derivation, to be compared with the previous step).

[3] In programming languages like Python, expressions are assigned addresses, which are accessed in the execution of a program by means of variables. It is even possible to access the memory address by means of a specific function, *id*(…). It is also possible to create *local variables*, which get created every time a function is called and erased when the function returns an output; this kind of operation would be analogous to the procedure to create *copies* in movement operations (since the copy is created and stored only temporarily, to be merged later in order to satisfy some featural requirement). In this case, we need two 'spaces': the 'shell' (where the program is executed) and the list of objects in memory (which contains the address, type, and value assigned to each variable). We will come back to the issue of addresses in **Section 3**.

all in properly formulating a theory of *copies* (vs. repetitions)? Does the workspace play any role in defining locality relations (can it, if it also comprises the lexicon)? In sum: what exactly does proposing the existence of a *workspace* buy us in theoretical and empirical terms? Let us focus on *copying*. Suppose that the workspace is the lexical array + the syntactic object being built. This does not help in solving the issue of determining 'where copies are stored temporarily'. If the workspace is a buffer or working memory, then this does constitute a potential solution, but this option is not spelled out as such in the literature; furthermore, now syntactic derivations would involve *three* spaces: the lexicon / lexical array, the phrase marker, and the memory buffer. In both senses of *workspace* the problem of having to define exactly how and where copies of syntactic objects are temporarily stored (among other issues) still arises. Interestingly, Collins & Stabler (2016) do note that

> *In minimalist literature, the term "workspace" is also used in a sense where two syntactic objects which are being built in parallel occupy two different workspaces. These two different workspaces are combined at some point in the derivation (see Nunes 2004: 140). We do not use the term "workspace" in this sense in our formalization. At any stage in the derivation there is only one workspace. Formalizing the alternative in our framework would not be difficult.*

However, that formalisation has not been made, to the best of our knowledge. Thus, the veil around the concept of *workspace*, what it is, and what it does (how it interacts with the operations of structure building and structure mapping in explicitly, fully worked out derivations), remains firmly in place.

The mention of a *workspace* in the formulation of the structure building operation (External Merge EM) is only in some sense a novelty, since it has been around (more or less explicitly) in several Minimalist accounts of structure mapping (i.e., Move / Internal Merge IM): 'moving' a syntactic object has been looked at in terms of Copy + Re-Merge of that object (see Chomsky, 2000; Nunes, 2004; Johnson, 2016 for a variety of perspectives). As observed above, the Copy-based approach has some problems which pertain to the lack of explicitness about the specific mechanisms involved in copying syntactic terms. Stroik & Putnam (2013: 20) formulate the issue very clearly:

> *To "copy X" is not merely a single act of making a facsimile. It is actually a complex three-part act: it involves (i) making a facsimile of X, (ii) leaving X in its original domain D1, and (iii) placing the facsimile in a new domain D2. So, to make a copy of a painting, one must reproduce the painting* ***somewhere*** *(on a canvas, on film, etc.), and to make a copy of a computer file, one must reproduce the file somewhere in the computer (at least in temporary memory).* (highlighting ours)

Note that this issue arises in both versions of *workspace* seen above: the workspace as a syntactic object or as the space where syntactic operations apply (and which properly contains the phrase marker being built). In addition to the problem posed by lexical selection and lexical insertion in the new definition of MERGE (Chomsky, 2019, 2020; who also does not provide a definition of *workspace*), new questions arise. Is the workspace co-extensive with the phrase marker or does it properly contain the phrase marker (and if so, what else is there?)? When a syntactic term is being copied (either within a single tree or across local trees, as in sidewards movement, see e.g. Nunes, 2004), the original position of this object and its target position are structurally distant (for some specific definition of *distance*; we will come back to this issue shortly), which also means there are derivational steps in between the introduction of these two positions: where is the copy kept active / accessible throughout this process? And, furthermore, if syntactic movement is triggered by the need to valuate / check / delete uninterpretable features on the moved element, the *goal* of Agree (and possibly also on the probe), how is the system capable of relating the pre-movement and the post-movement instances of the relevant syntactic object if their featural composition is different (Krivochen, 2015b: 266)?

This problem, the lack of explicit definitions and discussion about *where* syntactic operations apply (i.e., the lack of a systematic analysis of syntactic workspaces and the way in which they interact with, affect, or constrain operations), is not exclusive to the Minimalist *copy theory of* *movement* or the revival of Generalised Transformations in the original definition of Merge. It is important to point out that the problem of specifying *where* operations take place arises both in the case of EM and IM. Stroik & Putnam (2013: 21) point out that the distinction between EM and IM can be rethought within a *Copy-only* system in which differences are determined by the source and the goal of the *Copy* operation (Lexicon-to-phrase marker vs. phrase marker-to-phrase marker): this reworking of IM and EM, unlike the MGG version, makes it explicit that IM and EM differ in terms of the *spaces* that get accessed in each case and how the targeted syntactic object is affected –whether the space gets extended or not- (see also Stroik, 2009). EM is actually more complex than IM, in some (informal) sense (see also Chomsky, 2020: 23), as it involves a relation between *two* distinct spaces, and possibly a further operation of *selection* such that only some elements of the Lexicon are used in a given derivation (e.g., Chomsky, 2000: 101; also Chomsky, 2012: 3). This is a direct and (as far as we can see) unavoidable consequence of dissociating *lexicon* from *syntax* and these two from the 'interfaces' (the sound and meaning systems): a core assumption in lexicalist generative grammar is that syntax, semantics, and morpho-phonology are distinct components and that of these only the syntactic component 'produces' (an informal understanding of 'generates') structure. It is interesting to note that the operation *Transfer*, which takes syntactic domains and sends them to the interfaces, has been looked at from the perspective of *what gets transferred* (Chomsky, 2001; 2004; 2013; 2015); this is a crucial aspect of *phase theory*. Deciding which the phase heads are (Chomsky, 2000, 2001) and whether it is the complement of the phase head or the full phase that gets transferred (Epstein et al., 2015b; Bošković, 2016) have been major questions in the Minimalist agenda. However, little if anything has been explicitly said about *where* these syntactic objects are transferred *from* and whether the space *to* which they are transferred has the same properties as the source. In other words: are the interfaces (where objects are transferred to) isomorphic to the syntactic workspace? If so, why and how? If not, why not and in which ways?

In this paper we address some of the problems and questions that arise when the notion of a *workspace* is embedded in the context of syntactic theory with the purpose of aiding in assigning structural descriptions to natural language strings, and propose a way in which thinking about syntactic workspaces in terms of *topological spaces* (rather than stack tapes or memory buffers) has some important empirical and theoretical consequences. As observed before, work on topological properties of phrase markers (as topological graphs) although still uncommon in generative syntax has been carried out fruitfully since the mid-60s; here we attempt to continue that tradition in providing a formal approach to syntactic workspaces. We are primarily concerned with two aspects in the analysis of syntactic dependencies involving X, Y, W as in (4): (i) the *distance* between X and Y and W and Y in the definition of syntactic relations, and (ii) the properties of the *spaces* where these dependencies are defined. These concerns are (not so) implicit in the idea of *long-distance dependencies* and *discontinuity* in syntactic relations, a major topic in syntactic research. In this paper we will aim at defining *workspace* in a mathematical sense: the syntactic workspace will be defined as a *topological space*. In this context, we will focus on a topological interpretation of the notion of *distance* between elements in that space that we can use to shed new light on the problem of syntactic dependencies assuming, with MGG, that operations (in particular, Copy and Re-Merge, but also indexing and the determination of structural contexts for occurrences of syntactic objects) occur *somewhere*. However, we will depart from MGG in the characterisation of the operations themselves. In addition to providing an explicit definition of *workspace* and analysing the consequences of adopting it, the main innovation of our work consists on the proposal that computational operations transform the *workspace* rather than combine (Merging, concatenating, etc.) a set of discrete syntactic objects (features, lexical items, roots and categorisers…)[4].

2. Some properties of topological spaces

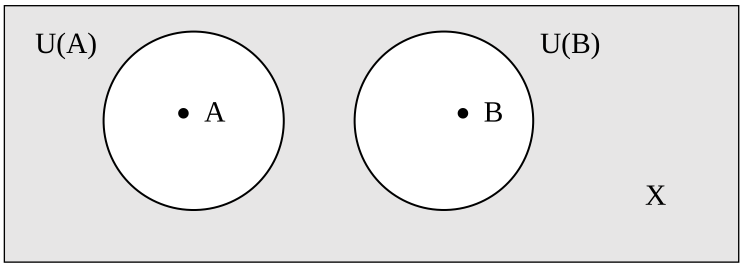
If we want the notion of workspace to work for us, theoretically and empirically, we need to provide an explicit characterisation of what a workspace is, what properties it has, and how it determines the kinds of operations that can apply in it and the format of objects that can serve as

[4] It must be borne in mind that we present *a* topological view of syntax, not the only possible one. The mathematical properties of spaces that we will describe here may be compatible to different extents with several versions of MGG including Chomsky's recent 'reformulation' of Merge (Chomsky, 2019, 2020; Chomsky et al., 2019; see also Epstein et al., 2020; Kato et al., 2016). In any case, the crucial point we want to make with this paper is that if the notion of *workspace* is to be more than a rhetorical device, some mathematical discussion is unavoidable.

inputs and outputs of those operations. In order to fully understand what assuming *workspaces* in syntax commit us to, some definitions are in order: as anticipated above, we will explore the possibility that syntactic workspaces are topological spaces, primarily because this would make available to us a set of mathematical tools that we can use in our inquiry. First, we need to introduce the concept of *topological space* (see Sutherland, 2009: Chapter 5 for basic notational and terminological points). A topological space is defined as *a set of points*, along with *a set of neighbourhoods for each point*, which satisfy a set of axioms relating points and neighbourhoods. This is a very general definition, and we need to get into some details.

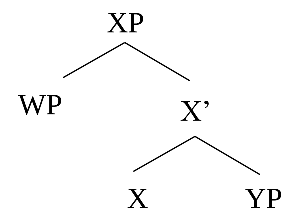
Let A and B be points in a topological space X. Then, we need to define the *neighbourhoods* of A and B, call them U(A) and U(B). If A is a point in X, the *neighbourhood* of A is a subset *U* of *X* that includes an open set *V* containing A (Reid & Szendroi, 2005: 108). In simpler terms, the neighbourhood of A in X is a set properly containing A where one can move some amount without leaving the set: there exists a non-zero number ε such that one is in the neighbourhood of A if one's distance to A is equal to or smaller than ε. If we now consider distinct points A and B in X, we can now define conditions pertaining to the relation between U(A) and U(B), which will define different kinds of topological spaces. A set is *open* iff it is a neighbourhood for every one of its points, and *closed* otherwise (if there is a boundary set, then the union of an open set and its boundary set defines a closed set). The distinction between closed and open neighbourhoods is essential, since bringing points closer together (thus affecting the *distance* function between them) can make their neighbourhoods intersect if these are open[5]. We can provide a graphical representation of disjoint closed neighbourhoods for A and B in a topological space X:

5)

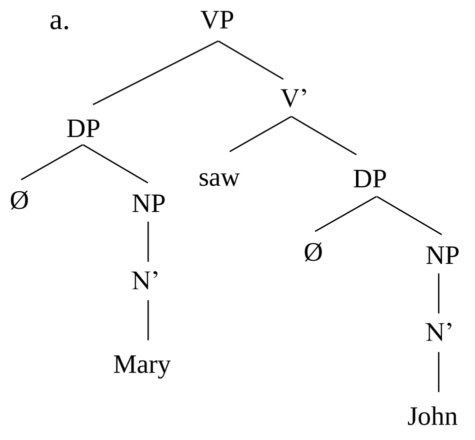


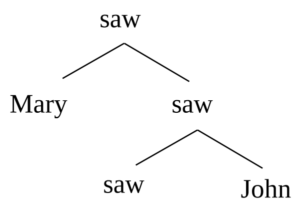
[5] We can choose between different axioms which relate points and their neighbourhoods, and the specific axioms that we choose gives us a classification of spaces. Furthermore, we can define functions that take us from one kind of space to another (for technical details, see e.g., Willard, 2004; Hazewinkel, 2001).

The notions of open and closed neighbourhoods should resonate with the syntactician; let us illustrate why. Consider a phrase marker like (6):

6)


In classical generative grammar, we would say that X 'projects' an XP, which is a syntactic object (a nonterminal node, in formal language theory) whose distribution and semantic interpretation are determined by the properties of X, the 'head' of the phrase. As has been observed repeatedly in the literature (see in particular Chomsky, 1994; Collins, 2002; Seely, 2006; Hornstein & Nunes, 2008; also Osborne et al. 2011 for critical discussion from a dependency grammar perspective) there are problematic issues (some empirical, most intra-theoretical) with the idea of 'intermediate' and 'maximal' projections that have motivated a departure from the XP-X'-X template used in X-bar theory and the adoption of so-called *bare phrase structure*: essentially, trees without extrinsic labels (like NP, VP, etc.), bar levels, or unary-branching. Under *bare phrase structure* assumptions, (7a) becomes (7b) (e.g., Chomsky, 1995: 226-228; Uriagereka, 2002):

7) a.


b.


These phrase markers, in Minimalism, would be built by means of EM in a workspace. However, without a clear definition of what a workspace is, we cannot even begin to ask how the properties of workspaces are related to the properties of the operations that can apply and the formal objects that can be constructed. If we adopt the idea that syntactic workspaces are topological spaces (and thus that phrase markers are topological graphs), we can ask what it means for XP to be the 'projection' of X from a different perspective which allows us to simplify the theoretical machinery and, as we will see shortly, has independent empirical advantages: using the definitions provided above, the projection of a head H can be characterised in terms of relations in the workspace as the *neighbourhood* of H. In other words: the *neighbourhood* of X contains WP and YP (which in turn are the neighbourhoods of W and Y, respectively, for X, W, and Y points in the workspace).

Let WS be our workspace, with X, Y, and W points in WS. Then, XP is a subset of WS. In (6), WP and YP are *interior sets* of XP, that is: W and its neighbourhood, and Y and its neighbourhood, are subsets of XP: we will see that Y and W differ in terms of whether their neighbourhoods are accessible from outside XP. This allows us to make the notion of 'projection of X' (and, as we will see shortly, the definition of elementary trees in a lexicalised grammar) into a concept that, configurationally, does not need independent definitions other than *point* and *neighbourhood*, which we need in order to define *workspace* anyway. This is a desirable consequence of the present view, as far as we can see, since properties of local derivational units can be defined in terms of the workspace. Note, incidentally, that we are assuming here that points in the workspace correspond to lexical items: this is not a necessary assumption. The approach to workspaces adopted here is compatible with different ways to define the atomic units of syntax: lexical items, features, constructions / signs, roots and functional morphemes, or basic expressions of the language (single-word or multi-word). A space is a set of points and a metric defined on them; there is no constraint to what 'points' are, in linguistic terms. The choice, as far as we can see, is eminently empirical and not dictated by the formalism.

What the definition of the 'projection' of a head as the *neighbourhood* of that head defined in the workspace allows us to clarify extends to relations within that projection: in particular, the concept of *edge* of a syntactic object becomes much clearer, and can be defined without stipulations (see Richards, 2011 for a discussion about the necessity of *edges* in Minimalist phase theory to avoid undergeneration). In an otherwise impenetrable domain (for example, a phase), the *edge* is a structural position that is accessible for operations triggered by a head external to that domain. What exactly constitutes the *edge* of a domain and is therefore (i) accessible for external operations and (ii) safe from Transfer has turned out to be a problematic issue. Accessibility is a central concept in contemporary generative theory, which is heavily based on operations over features: valuation, inheritance, sharing, donation (Chomsky, 2008; Ouali, 2010; Epstein et al., 2015a, b; Zeijlstra, 2020); to such an extent that other operations, like labelling (the old relation *is-a*), are seen as parasitic on Agree (Chomsky, 2013, 2015; also Zeiljstra, 2020). This of course represents a drastic departure from the framework of formal language theory that generative grammar was born out of, since labels no longer constitute a set of nonterminal nodes to be manipulated by rewrite rules, but are rather the output of other syntactic operations (Minimal Search, Agree, etc.). These operations rely on syntactic objects being *close enough* to establish a dependency: from the beginning of Minimalism, economy principles like Minimal Link and Shortest Move (Chomsky, 1995: 297) were aimed at reducing computational complexity in derivations by re-casting long-distance dependencies into sets of local chains (see Martin & Uriagereka, 2014 for discussion).

We need to consider the issue of *accessibility* in some more detail. Specifically, for instance, Chomsky's (2000, 2001) Phase Impenetrability Condition depends on how much the *edge* of a syntactic object comprises; in other words, what counts as *close enough* to the phase head to undergo Transfer:

> PIC$_1$ = *In phase α with head H, the domain of H is not accessible to operations outside α; only H and its edge are accessible to such operations* (Chomsky, 2000: 108)
>
> PIC$_2$ = *The domain of H* [a phase head] *is not accessible to operations at ZP* [the next phase]*; only H and its edge are accessible to such operations.* (Chomsky, 2001: 14)

Second-order conditions over operations like Agree (including varieties of Minimality; see e.g., Rizzi, 2016) also crucially depend on there being an unambiguous definition of *distance* between points in the space where the phrase marker is defined (an issue we will come back to). Counting the number of edges between objects A and B, where A probes for B (as in Kayne, 1984) requires taking the graph-theoretic view of phrase markers literally (such that structural descriptions are sets of *vertices* connected by *edges* -or *arcs*- where dependencies between syntactic objects are defined in *paths*). But even if we wanted to maintain the requirement that structural descriptions have only the format of binary-branching trees, the problem of defining the properties of the spaces where these trees are derived and how to define legitimate relations between distinct local domains in a way that does not require introducing additional principles (like the PIC itself) but rather follows from fundamental properties of these spaces remains unaddressed.

In this context, it is worth turning towards the question of whether there are properties of the 'workspace' which can deliver these properties: locality and accessibility. This is another point where the definition of local syntactic domains in terms of *neighbourhoods* of heads pays out theoretically and empirically. Recall that the neighbourhood of X may be *open* or *closed*. In this context, we can propose the following condition: if WP is accessible from outside XP, and WP is a syntactic object dependent of X, then we can define the neighbourhood of X, U(X), to be *open* and WP to be the *edge* of U(X). Crucially, it is possible for *interior* points of a set to be *limit* points as well: what we call syntactic *edges* are such points. *Complements,* on the other hand, would be interior points that are not limit points (of the neighbourhood of X in WS). This has far-reaching empirical consequences for the analysis of extraction phenomena, which have been studied in generative grammar for a long time. Chomsky (2019: 280) goes as far as equating *accessibility* in the workspace with *recursion*, although no argument is provided to this effect (but see Chomsky, 2020: 42, 56; accessibility is still restricted by stipulation since the definition of accessibility provided in that work can be applied recursively). We may speculate that this identification is due to the assumption that if something is accessible it may be used in future operations, but this leaves aside and unaccounted for the issue of lexical selection and the construction of lexical arrays. In other words, it side-steps the issue of how the system knows what elements it has to work with. The notion of *accessibility* is not formally defined in Chomsky (2019, 2020), Chomsky et al. (2019), and related works (other than the claim that '*if something was accessible to MERGE in WS it has to still be accessible in WS*'', in Chomsky, 2020: 56), but if we take the idea that operations apply in a workspace seriously, then we can provide an answer: *accessibility* is defined in terms of a syntactic object A being contained in an *open* *neighbourhood* if it is to be targeted by an operation triggered by a distinct syntactic object B. Closed neighbourhoods, thus, prevent accessibility. This is a crucial point in our proposal: we will propose that lexicalised grammars are a natural good match for a theory of workspaces based on points and their neighbourhoods. In particular, we will follow Joshi & Schabes (1990), Sarkar & Joshi (1997), and Frank (2002, 2013) in their formulation of a lexicalised grammar featuring *substitution* and *adjunction*.

2.1 Neighbourhoods and elementary trees: lessons from the Spanish auxiliary system

The concept of *neighbourhood* provides us with more than we may realise *prima facie* in terms of linguistic analysis: in phase theory it is necessary to identify designated nodes as 'phase heads', and cyclicity is driven by the opacity effects defined by the PIC; however, since phases are defined purely in terms of syntactic properties (being probes for Agree), there is currently no semantic (propositional) or lexical motivation for phasehood in the Chomskyan version of the theory (see e.g., Chomsky, 2008; Bošković, 2020). However, this need not be so: if syntactic cycles are structured as the neighbourhood of lexical heads (as opposed to being the complement of functional heads, as in phase theory), then we can use the concept of *workspace* and the auxiliary notions it forces us to define to our advantage, pursuing a more descriptively adequate theory of the syntax-semantics interface. The result of adopting the assumption that the units of the grammar are structures with a single lexical anchor is a *lexicalised* grammar: being able to define the neighbourhoods of lexical heads in the syntactic workspace makes lexicalised grammars a natural way (but not the only one) to implement the theory of workspaces that we put forth here. Concretely, we have in mind a proposal like Frank's (1992, 2002), in the context of Tree Adjoining Grammars (TAGs):

> *Condition on Elementary Tree Minimality* (CTEM)*: Each elementary tree consists of the extended projection of a single lexical head* (Frank, 1992: 53)
>
> *The syntactic heads in an elementary tree and their projections must form an extended projection of a single lexical head.* (Frank, 2002: 22)

The appeal of TAGs for a syntactic approach that takes workspaces seriously is hard to overstate. In a TAG, the grammar generates a set of *elementary trees* (initial and auxiliary trees), which are combined to yield *derived trees* by means of two operations: *substitution* and *adjunction* (Joshi, 1985). The aspect of TAGs that becomes particularly relevant under the present assumptions is that the development of TAGs found in Frank (2002, 2013) and Joshi & Schabes (1990), known as *lexicalised TAGs* (LTAGs) allows for a lexically-semantically motivated definition of local domains. This is so because elementary trees, which are the basic units that are manipulated by the grammar, are defined as in the CTEM. In Krivochen & García Fernández (2019) we build on Frank's CTEM and define an elementary tree as the minimal unit that contains the following elements:

8) a. A predicative basic expression *p*
b. Temporal and aspectual modifiers of *p*
c. Nominal arguments of *p*

This definition of elementary trees as the basic units of the grammar (both in terms of syntactic composition as well as semantic interpretation) results in a strongly local syntax, in particular if we add the requirement that

Every syntactic dependency is expressed locally within a single elementary tree (Frank, 2013: 233)

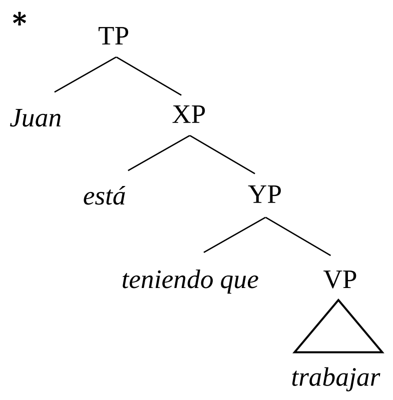
The definition of workspace proposed here allows us to capture the insights of LTAGs: elementary trees can be defined in terms of the workspace as the neighbourhoods of lexical heads. A major empirical advantage of LTAGs is that, because they are not committed to uniformly monotonic antisymmetric phrase markers, it is possible to define structures where semantic dependencies are better represented. We can briefly exemplify the descriptive power of such a definition of local domains with dependencies between auxiliaries in Spanish. Bravo et al. (2015) and Krivochen & García Fernández (2019) argue, on empirical grounds, that a monotonic, Merge-based phrase structure grammar provides inadequate structural descriptions for sequences of auxiliary verbs in Spanish, and that considering the notion of *neighbourhood* in syntactic terms is empirically fruitful. Consider in this respect the contrast between (9a) and (9b):

9) a. Juan tiene que estar trabajando
J. Aux.Mod.deont Aux.prog working
'J. must be working'

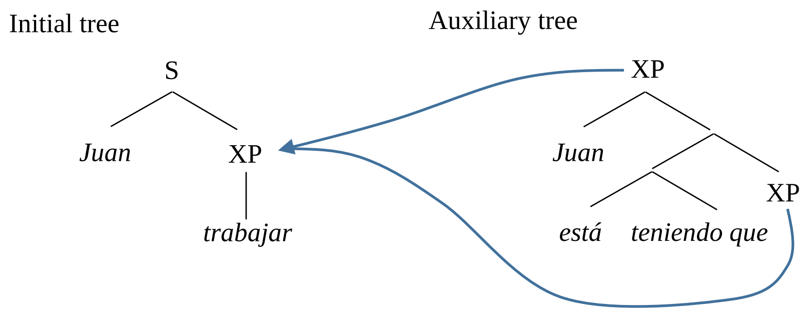
b. Juan está teniendo que trabajar
J. Aux.prog Aux.Mod.deont work
'J. is currently having to work'

Both (9a) and (9b) are grammatical and acceptable sentences in Spanish. Crucially, however, they are *not synonymous*: in (9a) the obligation pertains to a current, progressive event of working; in (9b) the progressive affects the obligation, but not the event of working. Therefore, (9b) does *not* entail 'John is working', since the progressive only modifies the modal, but not the lexical verb. In the aforementioned works, it is proposed that this is so because *estar* does not have scope over *trabajar,* which in phrase structural terms means that *estar* cannot c-command *trabajar* (May, 1985 and much related work). Furthermore, there is no evidence that either order is derived from the other via movement or any other kind of structure mapping (i.e., there is no empirical evidence to support the idea that *estar* moves from a position underneath the modal *tener que* to reach its surface position, as would be required by an approach to sequences of auxiliaries as in Cinque, 2004): both are equally 'basic'. Assigning a structural description like (9b') to (9b) (as would be predicted by antisymmetry-based MGG) is thus empirically inadequate, since in (9b') *estar* has scope over both *tener que* and *trabajar* (labels are inconsequential at this point):

9b')



In order to get the correct reading for (9b), the progressive auxiliary *estar* must be in the neighbourhood of the modal, but *not* on the neighbourhood of the lexical verb. In other words: it must be possible to define a syntactic unit that includes *está teniendo que* and excludes *trabajar*. This basic condition (required for descriptive adequacy) can be satisfied if the modal *tener que* heads its own elementary tree, which excludes (and therefore defines a distinct neighbourhood from that of) the lexical VP of *trabajar*. Therefore, we need two elementary structures, simplified along the lines of (10) below (see also Krivochen & García Fernández, 2019). The crucial aspects here are two: first, that the elementary tree headed by the auxiliary *tener que* gets adjoined to the elementary tree headed by *trabajar* (called the *initial tree*): the auxiliary (in the progressive) modifies the lexical VP and not the other way around. Second, that in order to implement adjunction in a TAG, the elementary tree that gets adjoined (called *auxiliary tree*) must have identically labelled nodes in its root and frontier *and* also identical to a node in the initial tree (Joshi, 1985). In this way, the node in the initial tree can be replaced by the adjoined structure, as schematised in (10):

10)


As we said before, PS-trees are used as expository devices; what is crucial here is that in order to provide a strongly adequate structural description for (9b) it is necessary to define a syntactic object where *estar* belongs to the neighbourhood of *tener que* (which we called XP) and this object excludes *trabajar*. The computational system must incorporate a way to bring those neighbourhoods together, which amounts to having an operation to compose derived trees: in TAGs this comes in the form of *substitution* and *adjunction*. Both are mechanisms that replace a designated node in an elementary with the root of another elementary tree under identity: *substitution* corresponds to the case where the designated node is in the frontier of the target tree (this corresponds to Chomsky's 1955 *generalised transformations*), and *adjunction* corresponds to the case where the designated node is in an internal (non-frontier) position.

There is another important issue: if both elementary trees contain a nominal argument (here, *Juan*), what happens when they undergo adjunction? Having nodes be points in the space becomes particularly useful here: *Juan* in the initial and auxiliary trees denotes the same entity, it is the defined by the same position in the workspace. Unlike Merge-based computation, where if we Merge two complex syntactic objects {Juan, {está, teniendo que}} and {Juan, {trabajar}} the result is {{Juan, {está, teniendo que}}, {Juan, {trabajar}}, a workspace-based lexicalised grammar can not only define local derivational units in terms of neighbourhoods of lexical predicates in the workspace, it can also identify all common nodes between the neighbourhoods of distinct lexical predicates and unify them into a single object (cf. Martin & Uriagereka's 2014 *chain collapse* mechanism, or Krivochen's 2015b *token collapse*). In order to do this, however, the condition that a node may have only one mother must be ditched, thus yielding a graph with *multidominance* (see also Citko, 2011; Johnson, 2016)[6].

A lingering issue is that saying that modal auxiliaries define their own neighbourhoods entails that they must be lexical predicates, according to the three requirements (8a-c) above. We need a definition of what exactly it means for an element to be a 'lexical predicate'. Bravo et al. (2015) and subsequent works argue that the crucial property to be taken into consideration is whether a predicate can both modify another expression and be modified itself or only modify. Let us expand upon this. That being an auxiliary verb is a matter of gradience is something widely acknowledged, mostly

[6] If elementary trees were defined as feature structures, the process we describe would be akin to Unification as defined in Shieber (1986: 14):

> *In formal terms, we define the unification of two feature structures D' and D" as the most general feature structure D, such that D' ⊆ D and D" ⊆ D. We notate this D = D' ∪ D".*

Some further clarification is necessary here. Within Unification grammars, ⊆ is used to symbolise a *subsumption* relation between feature structures, in which a feature structure, abbreviated D, contains *part* of the information of another D', such that D' ⊆ D. In more complex terms, the concept of *subsumption* is based on that of *dom(D)* (the *domain* of a feature structure, namely, the features it includes, regardless of their mapped values), such that D' ⊆ D *iff* ∀(*x*) | *x* ∈ *dom*(D'), *x* ∈ *dom*(D). A more concrete example will clarify the differences between *Unify* (which reminds of set-theoretic *union*, but binary) and (Chomskyan) *Merge* ((i) is taken from Jackendoff, 2011: 276, ex. 10 a, b):

i) a. Unification of [V, +past] and [V, 3 sing] = [V, +past, 3 sing]
(not *[[V, +past] [V, 3 sing]]*, as with Merge)
b. Unification of [VP V NP] and [V, +past] = [VP [V, +past] NP]
(not *[[V, +past] [VP V NP]]*, as with Merge)

among functionalists and typologists; within the generative tradition the category “semi-lexical” verb has been proposed in order to give account of verbs which share properties with both lexical verbs and auxiliary verbs (e.g. Cardinaletti and Giusti 2001). In Bravo et al. (2015) and Krivochen & García Fernández (2019) we propose that modals (epistemic, deontic, dynamic) and aspectual phasal auxiliaries belong to the class of auxiliaries that can both modify (a lexical verb or another auxiliary) and be modified by an auxiliary; we call them *lexical auxiliaries*. In contrast, perfective *have*, temporal *ir a* and *acabar de*, progressive *ser*, and possibly passive *ser* belong to the class of auxiliaries that can only modify (a lexical verb or a lexical auxiliary), but not be modified; we call them *functional auxiliaries*. We can exemplify these cases by considering the relations between auxiliaries in the following examples:

11) a. Juan tiene$_{\text{[Modal]}}$ que haber$_{\text{[Perf]}}$ empezado$_{\text{[Phasal]}}$ a trabajar
J. has that to-have started to work
‘John has to have started working’

b. Juan va a$_{\text{[Temporal]}}$ poder$_{\text{[Modal]}}$ empezar$_{\text{[Phasal]}}$ a trabajar allí
J. goes to can to-start to work there
‘J. will be able to start working there’

c. Debe$_{\text{[Modal]}}$ haberlo$_{\text{[Perf]}}$ hecho
May to-have=CL done
‘He/she has had to do it’

d. Va a haber estado trabajando
Goes to have been working
‘He/she will have been working’

In (11a) the lexical verb *trabajar* is *not* modified by *tener que*, only by *empezar*; perfective *haber* in turn modifies only *empezar*. Specifically, the obligation (realised by *tener que*) pertains to *start* working, not to the whole event of working. The corresponding segmentation must be [tiene que [[haber empezado a] [trabajar]]]. In (11b), the temporal auxiliary *ir a*, contributing future tense, modifies the deontic auxiliary *poder*: only the deontic possibility is located in the future. This possibility pertains to the starting point of the (habitual) event of working, denoted by the auxiliary *empezar a*. The correct segmentation for (11b) would then be [[[*va a poder*] [*empezar a* [*trabajar*]]]. The case in (11c) presents a modal that takes as a complement a perfect infinitive; this means that the perfect auxiliary is a modifier of the lexical verb, but is not modified itself by the modal: the lexical verb in the perfect is. The segmentation is [debe [haberlo hecho]], and not (as would be predicted by a Merge-based system) [debe [haberlo [hecho]]]. Finally, in (11d) there is a sequence of auxiliaries none of which is modified by the auxiliary to its left. Note that, because perfective active participles have an anteriority interpretation, if the future *va a* modified *haber* there would be a clash between a future temporal interpretation and a preterite temporal interpretation. Bravo et al. (2015) suggest that in such a case, the three auxiliaries modify directly the lexical verb without modifying any of the other members of the auxiliary chain. This is only possible if we define the neighbourhood of the lexical verb to contain these auxiliaries, but none of these auxiliaries can define a neighbourhood themselves[7]. In this sense, the syntactic implementation of workspaces departs from the topological definition: not all points are equal, and this is due to the presence of semantic factors. Neighbourhoods in syntax are not only formal topological objects, they are subjected to requirements of semantic interpretation and externalisation. If we define neighbourhoods of points in the workspace by the presence of (8a-c) above, the Spanish facts presented in this section receive an account that requires no additional stipulations: all syntactic and semantic dependencies are indeed established within elementary trees, because these correspond to the neighbourhood of lexical predicates.

This brief discussion serves the purpose of illustrating the relevance of the concept of the *neighbourhood* of a lexical head: this allows us to have derivations which do not grow at a constant rate and get chunked into smaller domains which cannot be defined *a priori* (as they depend on the presence of a lexical predicate). In a workspace-based lexicalised grammar, the units of computation are the neighbourhood of lexical heads, which we identify with TAG's *elementary trees* (Joshi & Schabes, 1990; Frank, 2002, 2013) but with a twist: in the version of TAGs used in Krivochen & García Fernández (2019) only *lexical predicates* define a neighbourhood, with lexical arguments and

[7] This distinction can be captured by a variety of formal mechanisms. For example, Krivochen & Schmerling (2020), working within a Montagovian categorial grammar, distinguish between expressions which are introduced by means of operations of *concatenation* and expressions introduced by means of *functional modification* (which take an expression of category X/Y and output an expression of category X//Y, which is a modified expression of category X/Y). The former correspond to lexical heads; the latter, to what Bravo et al. (2015) call *functional auxiliaries*.

functional modifiers (predicates that can only modify, but not be themselves modified) being part of the neighbourhood of lexical predicates. The notion of *metric space* allows us to formulate an explicit definition of *workspace* which is instrumental in defining a framework where locality and accessibility can be linked to properties of the spaces where derivations take place. Furthermore, it allows us to have a characterisation of syntactic terms in that space: as in Sarkar & Joshi (1997) (based on Gorn, 1967), we can define for each element in the neighbourhood of a lexical predicate (and the lexical predicate itself) an *address* which corresponds uniquely to a region of the space[8].

2.2 Metric spaces and distances in syntax

We have defined what a space is, in a very general way: a set of points and a set of neighbourhoods, plus a set of axioms allowing specific relations between these. Different axioms define different spaces, and since the core idea of this paper is that the properties of the workspace determine the kinds of syntactic objects that can be built in that workspace, we must get into some detail about *types* of spaces. Intuitively, points in a space can be close or far apart to different degrees: we can call the function that defines just how close or far apart points are the *metric* of the space. Here we have a crucial difference between two kinds of spaces (Kaplansky, 1977; Sutherland, 2009):

12)

For x, y, z points in a topological space S, S is ***metric*** iff:	For x, y, z points in a topological space S, S is ***ultrametric*** iff:
a. $d(x, y) > 0$ if $x \neq y$ (*positive property*)	a. $d(x, y) > 0$ if $x \neq y$ (*positive property*)
b. $d(x, y) = 0$ *iff* $x = y$ (*identity property*)	b. $d(x, y) = 0$ *iff* $x = y$ (*identity property*)
c. $d(x, y) = d(y, x)$ (*symmetric property*)	c. $d(x, y) = d(y, x)$ (*symmetric property*)
d. $d(x, z) \leq d(x, y) + d(y, z)$ (*triangle inequality*)	**d. $d(x, z) \leq \max\{d(x, y), d(y, z)\}$ (*Ultrametric inequality*)**

We have suggested above that *distance* is a crucial notion in theoretical syntax; now we will see how a formal definition of *workspace* in topological terms can help us capture the theoretical insights.

[8] In a lexicalised grammar like the one assumed here, the distinction between lexical and functional categories is not one of featural composition or anything of the sort, but rather of the size of its neighbourhood: because neighbourhoods of lexical heads contain the lexical head, plus functional modifiers (aspectual and temporal auxiliaries), and the arguments of the lexical head, the neighbourhood of a lexical head is a superset of the neighbourhood of the same element in a functional use (e.g., Spanish *ir* 'to go' as a lexical unaccusative verb vs. the auxiliary <*ir a* + infinitive>, the periphrastic future form; Italian *avere* -lexical transitive- vs. <*avere* + participle>, a perfective auxiliary). See also Manzini & Savoia (2011) for related discussion.

The distance *d* over a set X is a function defined on the Cartesian product X × X (thus producing a set of ordered pairs of members of X); *d* will be called a *metric* iff the properties (a-c) plus the triangle inequality hold. The *triangle inequality* is a crucial property: it determines that distances in metric spaces *sum*: informally, if A is *m* away from B and B is *n* away from C in a line defined in X, then A is *m* + *n* away from C. This is an essential property of metric spaces, because it allows us to formulate the notion of *closeness* in comparative terms, such that *A* is *closer* to *B* than *C* if *d*(*A*, *B*) < *d*(*B*, *C*) (i.e., if *m* < *n*). What *distance* is measured in terms of may vary (criteria to measure *distance* include: total number of nodes, number of cyclic nodes, number of nodes and edges, number of potential governors for a specific syntactic relation…), but its importance cannot be denied for the theory of syntax. Furthermore, distances *sum*, as per the triangle inequality. The fact that distances between points vary and sum (as per the *triangle inequality*) distinguishes metric spaces from *ultrametric* spaces. Gajić (2001: 96) provides a simple characterisation of ultrametric spaces:

Remark: *Let* [a topological space] *X ≠ Ø,* [with a] *metric d defined on X by*

$$d(x, y) = \begin{cases} 0, if\ x = y \\ 1, if\ x \neq y \end{cases}$$

[then, the] *so-called discrete metric is ultrametric*

Since distances between points are either 0 or 1, ultrametric spaces preserve topological distinguishability but all distinct points are equidistant (at a distance 1 from each other). One of the main things that we want a formalisation of workspaces to do for us is capture the idea that: two *distinct* points A and B in X can be arbitrarily near or far apart (however *distance* is calculated, more on this below), but never have 0 distance. A and B can only have a 0 distance *iff* A = B; this is crucial for our purposes. In other words, we want a space that *preserves topological distinguishability* and in which variable distances can be defined unambiguously. Here we will propose that an adequate characterisation of syntactic structures requires being able to sum distances and determine, given syntactic objects X, Y, Z in a structural description, whether X is closer to Y than Z. Therefore, *metric* spaces become the natural option as a characterisation of syntactic workspaces.

The properties of ultrametric spaces, while not viable for syntactic structural descriptions, do make thinking about the *lexicon* easier: after all, not only do we have topological distinguishability in ultrametric spaces, but also the fact that distances do not sum entails that we can think of the lexicon as an associative network in which there are no biases: anything can, in principle, be selected and connected to anything else. Once elements from the lexicon enter into syntactic relations (and only then), we can define variable distances in the context of a phrase marker: here is where elementary trees are built. Following this reasoning, if the lexicon needs to maintain distinguishability but not have pre-encoded biases (in the form of distances), then we can characterise the lexicon as a set of elements with an ultrametric metric (following Saddy, 2018). When a set of items enter syntactic relations, this determines variable distances between them, which disrupts the ultrametricity of the lexicon. The argument exposed in Saddy (2018) and Krivochen (2018a) is that a formalism defined in an ultrametric space has no conditions that can prevent the generation of local syntactic structure (e.g., a set of lexicalised elementary trees) because there are no heads or edges, given the ultrametric inequality. However, if the space is metricised, then we can define lexical heads and their neighbourhoods, which constitute the atoms of syntax in a lexicalised grammar. It is important to note that in this perspective there are not *two spaces*: lexicon and syntax. Rather, structure emerges when the ultrametricity of the lexicon is disrupted by syntactic operations. Properties of the space where a phrase marker is defined, then, gives us some hard constraints with respect to the properties that can be ascribed to such construct. This has major consequences for the kind of grammatical formalism that can be formulated in each space: on the one hand, we can enjoy the advantages of ultrametric spaces as a characterisation of the initial state of syntactic derivations (topological distinguishability, constant distances, no biases towards specific relations between elements); on the other hand, when the ultrametric space is metricised, we can define open neighbourhoods of lexical heads: these are elementary trees, the building blocks of syntax. In the metric space, we can define variable distances between lexical predicates and their neighbourhoods.

Why would we bother with this? Because *accessibility* and *distance* are fundamental concepts in generative grammar, particularly in Minimalism. Closer dependencies are favoured over longer dependencies: Kayne's (1984) take on the Empty Category Principle, Chomsky's (1995) Shortest Move and Minimal Link Condition economy principles, Chomsky's (2013, 2015) Minimal Search; Rizzi's (2016) Relativised Minimality, to give a few examples, crucially depend on there being a way to determine, given syntactic objects A, B, and C, whether a 'minimal' operation (e.g., MERGE, Simplest Merge, Agree, etc.) can establish a dependency of some sort (labelling, probing, theta-marking, etc.) between A and B or A and C, or whether B and C are equidistant from A. Syntactic operations over features or feature bundles may thus be constrained in terms of how much structure is there available to *probe into*; if such operations apply to syntactic objects in a workspace, then defining a metric on that space becomes an important part of an adequate meta-theory for these syntactic operations.

On the issue of accessibility, let us go back to the formulations of the PIC above. We can compare the probing space that each of them allows for:

13) [$_{CP}$… [C [$_{TP}$… [T [$_{vP}$ Ext Arg [*v* [$_{VP}$… [V Int Arg]]]]]]]]

In (13), the search space for an operation triggered by T is only Spec-*v* according to PIC$_1$, but all the way to the complement of V according to PIC$_2$ (since T is not a phase head). The extra probing space in PIC$_2$ seems to be required for the movement of VP internal subjects (unaccusative / ergative subjects) to Spec-TP, triggered either by a feature in T before the merger of C or by a feature *inherited* by T from the phase head C. In any case, and empirical issues notwithstanding (e.g., pertaining to the choice of phase heads, the issue of whether more than one specifier position is indeed allowed, etc.), it seems clear that (a) the notion of *distance* is essential in the formulation of syntactic operations and conditions over these, and (b) if this notion is to be implemented in a system that also assumes the existence of a *workspace*, then it stands to reason that *distance* be defined in terms of properties of the workspace: if it is to be formulated in terms of elements in a phrase marker (number of nodes or edges between relevant syntactic objects), then these must be characterised in terms of the workspace. An alternative is to define *distance* without making reference to the workspace. This is the road taken by recent research on labelling, based on Chomsky's (2013; 2015) suggestion of a Minimal Search algorithm, which given a syntactic object finds a head with suitable features for identification at the interface levels. In the next section we will briefly look at some problems that arise with Minimal Search as characterised in the literature, and how these problems could be circumvented with the use of some basic mathematical tools that are put at our disposal in our definition of workspace.

2.2 *Distances and Minimal Search*

It is remarkably difficult to find an explicit formal definition of Minimal Search (for example, Chomsky, 2013, 2015, 2019, 2020 provides none), partly because it is characterised as a 'third factor property' (e.g., Chomsky, 2013: 39; Epstein et al., 2015a: 8, 13), which are mysterious in their own right[9]. If Minimal Search is indeed an algorithm, it must be similar to algorithms commonly used in search trees. Of these, we will consider the simplest ones: *breadth first* and *depth first*.

Traditionally, Merge generates binary-branching, single-rooted, labelled trees (Chomsky, 1995; Kayne, 1994). These trees are, mathematically, finite sets of nodes and edges: they are *graph-theoretic objects*. It is important to note that at this point we are forced to adopt a graph-theoretic perspective, because it makes no sense to ask for the *distance* between elements in a set. For purposes of Minimal Search, in particular applied to labelling, we are concerned with finding the shortest path between two nodes; assuming that labelling is not an operation triggered by a head, the 'search' should start from the root (which would be unlabelled after Merge). What we are interested in, then, is the distance between the root, r, and the first head H; we notate this as $d(r, \text{H})$. 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 *depth-first search* also starts from the root, but instead of going through all the nodes in a generation before proceeding to the next, it explores the leftmost branch

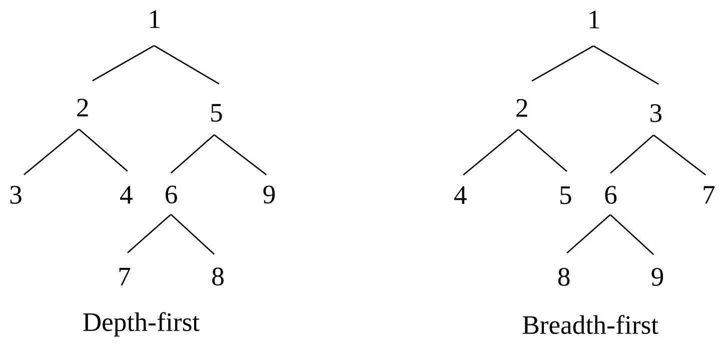
[9] Chomsky (2020: 48) characterises Minimal Search as follows:

> […] *the third factor principle Minimal Search MS, a process that terminates once it reaches the head of a chain (a principle that has good empirical motivation, which I'll have to put aside here).* […]

This characterisation poses more questions than it answers. How does the system know that something is the head of a chain? Do chain-heads bear some sort of feature that identifies them as such? If so, why wouldn't that also be available for other versions of Merge, which Chomsky claims pose problems, like Parallel Merge?

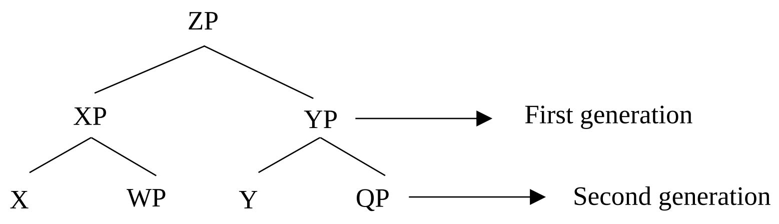
exhaustively before backtracking and proceeding with the immediately adjacent branch (Even & Even, 2012: 11, 46-49; Cormen et al. 2001: 531, ff.). We can illustrate the order in which nodes are visited in a simple tree for both search algorithms:

14)



However, the arguments found in Epstein, Kitahara & Seely (2020) (which expand on Chomsky's, 2013, 2015) suggests to us that Minimal Search is neither a *depth first* or a *breadth first* search algorithm: we know this because the analysis of {XP, YP} situations proposed in the current Minimalist literature characterises them as 'ambiguous' for Minimal Search, requiring some modification of the symmetric object in order to yield an unambiguous result. Consider the following phrase marker:

15)



In a *depth-first* search, starting from the root, the algorithm would visit nodes in the following ordered sequence Σ:

Σ = <ZP, XP, X, WP, YP, Y, QP>

such that X is visited before Y. In a *breadth-first* search, the sequence of nodes would be the ordered sequence Σ':

Σ' = <ZP, XP, YP, X, WP, Y, QP>

Again, X is visited before Y. In neither case would the search be 'ambiguous': if you know where you start from, the algorithm would find one of the heads first. Furthermore, there is no standard search algorithm we could find that would locate both heads simultaneously, which is the only way in which labelling would fail or movement would be ambiguous (Chomsky, 2013: 43; Epstein et al., 2020: 2; Carstens et al., 2015). The only possible scenario in which both heads are located simultaneously is if the search proceeds from the root *at the same through two edges*. Simultaneous operations, however, are not characteristic of formalisms in the Chomsky hierarchy (as opposed to cellular automata, for example, Lindenmayer grammars; see Prusinkiewicz & Hanan, 1989: 5), which follow a so-called *traffic convention*: rules are applied sequentially and one symbol at a time. Allowing the grammar to carry out two search probes at the same time (one for the path ZP, XP and another for the path ZP, YP in (15)) implies a departure from basic conditions of grammars in canonical form that requires an explicit formal argument not provided in the literature. Similarly, under present assumptions, it is not possible to say that in a configuration like ... $[_T$ T $[_\alpha$... DP... *v*P]] '*D and V-v should be equally available for a relation to T, since both are equally close to it* […] *Minimal search should find D and v equidistant from T*' (Carstens et al., 2015: 71). Equidistance of X and Y with respect to a probe Z in a phrase structure tree is possible either in a non-metric space (specifically, an *ultrametric* space) or if Z is the mother node of X and Y. Any other configuration would entail, given *breadth first* and *depth first* algorithms, that one of X or Y is found first. The core issue, as far as we can see, is that there is no proper definition of Minimal Search or *distance* or a proper characterisation of the kinds of objects to which Minimal Search applies: *distance* can receive a straightforward definition in graph-theoretic terms, but we must bear in mind that in the orthodox perspective, Merge creates sets, not graphs (see e.g. Chomsky, 1995, 2013, 2020; Collins, 2017; Collins & Groat, 2018; Epstein et al., 2015a, b; among many others). Defining *distance* in terms of sets is required in current Minimalism, but to the best of our knowledge it has not actually been done. This is due, we believe, to the fact that trees are rarely analysed as mathematical objects in MGG; in this context, the concept of workspace (where trees would be built) remains arcane.

We are well aware that there are many issues that we have either ignored or touched upon only superficially for reasons of scope and space, the point that we want to make is simply that there is much to win by formalising the properties of the spaces where these objects are created, which in turn forces upon us a limited set of possible structures that can arise. In the view pursued here, the syntactic workspace is a metric space, where the units of the syntactic computation are elementary trees defined by the presence of a lexical predicate. In **Section 3** we will address the issue of how these structures are put together, since so far we have defined a set of elementary structures (the neighbourhoods of lexical predicates) and hinted at the existence of composition operations but we have not operationalised them.

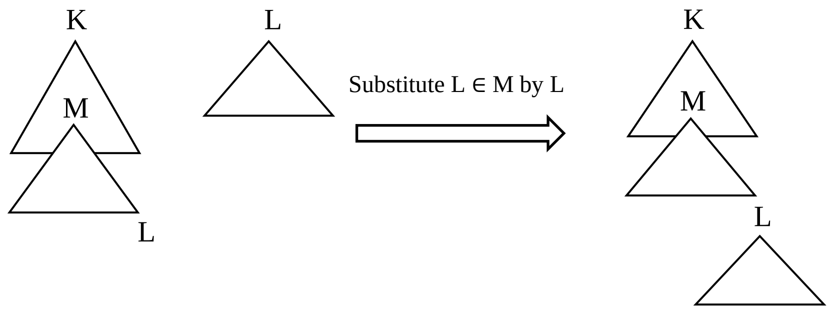
3. Syntax in the workspace

We have a characterisation of the workspace where syntactic operations apply and, in that space, we have the elements that appear in structural descriptions defined as points or sets thereof within said space. This metric space is populated by a set of 'elementary trees' (a term we keep for expository convenience only, since they are not technically trees), defined as the minimal unit that contains a lexical predicate, its functional modifiers, and its arguments: these are graphs each of whose vertices (or 'nodes') is a basic expression of the language. Elementary trees are the neighbourhoods of lexical predicates. If neighbourhoods are open, then elements within one neighbourhood become accessible from the other neighbourhood: this allows for the composition of local neighbourhoods and the formation of complex structures (TAG's *derived trees*). In this context, consider the following condition on a strongly cyclic syntax, from Uriagereka (2012: 75):

> *Whenever a phrase-marker K is divided into complex sub-components L and M* […], *the daughter phrase-marker M that spells-out separately must correspond to an identical term M within K.*

Uriagereka's requirement proposes a solution to the problem that arises in all models of cyclic structure interpretation: once an input has been chunked and each part has been subject to an arbitrary set of operations, how to we put everything back together? From a generative-derivational viewpoint, in which structure is built step-by-step by means of discrete recursive combinatorics (e.g., the operation Merge), the question can be phrased as: how can separate *command units* (local monotonically derived phrase markers) be linked? The problem, when asked specifically about linguistic structures, pertains to the relation between strict locality in syntax and compositionality in semantics: if we adopt a lexicalised grammar, as in **Section 2**, we need both locality and compositionality in order to define local elementary trees (the neighbourhoods of lexical heads), assign local interpretations, and account for how these local units and their interpretations are combined. What we want to do is provide a way to capture structure composition *without having to invoke additional structure* in the form of semantically vacuous non-terminal nodes. A simple case we can deal with is that of *substitution* (Chomsky, 1955; Joshi & Kroch, 1985). In traditional phrase structure terms, let K be a term, and let M be a term within K, with a node L in its frontier. Furthermore, let L be the root node of a distinct term (not a part of K). Then, we can substitute the node L in M with the sub-tree (term) L:

16)



We have been deliberately imprecise with respect to which L we are referring to in each instance, the reason being that *substitution* works if and only if L in M is *identical* to the root of the separate term L (see Frank, 2002: 17, ff. for discussion). This is a simple case, which can correspond to clausal complementation:

17) John wished [that Mary would date him]

(17) contains two clausal domains in a hypotactic relation; each of which corresponds to an elementary tree by virtue of containing a single lexical predicate. The bracketed clause, whose lexical anchor is the verb *date* (call it L) is subordinated to the elementary tree headed by the verb *wish* (call it M). The elementary tree of *wish* contains a subject and a placeholder in its frontier that corresponds

to the direct object of *wish* (Chomsky, 1955, 1995); the elementary tree of *date* contains its arguments (a subject and a direct object) and a functional modifier in the form of the auxiliary *would*. Then, a derivation of (17) using *substitution* goes along the following lines (cf. Uriagereka's citation above):

18) a. [$_M$ John wished [L]]]

b. [$_L$ that Mary would date him]

c. [$_K$ [$_M$ John wished [$_L$ that Mary would date him]]] (via *substitution* targeting L)

In this case, what we have done is link the neighbourhoods of two points, *date* and *wish*, by identifying the same node in both syntactic objects: the address we have called L in M (16a) points to the same object as the address L in (18b): the root node of the sub-graph that is a structural description for the string *that Mary would date John*. What kind of objects are we dealing with? In the present conception, these are graphs whose vertices are nodes in the workspace. Defining a parsing of such a graph entails defining a walk through that graph: a *v_1-v_2 walk* in a graph G is a finite ordered alternating sequence of vertices and edges that begins in v_1 and ends in v_2. Further conditions can be imposed over walks. For example, we might require that

- each edge be walked on only once, or
- that each vertex be visited only once, or
- that both of these conditions hold, or
- that neither of them hold

In MGG nodes are multiplied in structural descriptions because both conditions hold. This is not necessary, though (see Krivochen, 2018b for extensive discussion). Whereas MGG defines *paths* in structural descriptions (Kayne, 1984), theories that allow for multidominance can define *trails*: a *path* is a walk in which no *vertices* are repeated, whereas a *trail* is a walk in which no *edges* are repeated, but *vertices* can be. This is crucial, since allowing a node to be visited more than once in a walk through a graph simplifies things, provided that nodes are appropriately defined. A well-defined walk through G must impose an order between nodes, such that $v_1 < v_2$ in G *iff* v_1 is visited before v_2 in a walk through G (this is a definition of *dominance* in G). Ojeda (1987: 254) proposes the following condition, which is a *total* order over nodes in a graph:

<u>Full Leaf Order</u>*: For all leaves u, v,* [either] *u < v or v < u.* (Ojeda, 1987: 258)

Because graphs are *strictly* ordered[10], and a strict order is transitive, the derived structure (18c) is also strictly ordered (Ojeda, 1987; Krivochen, 2018b). But we still have a problem: how do we get the relation *John-him* to hold? In an MGG-style phrase structure grammar it is necessary to invoke additional devices (e.g., indexing, plus a level of representation where indices are identified and interpreted) which allows the grammar to identify *John* and *him* as NPs which are assigned a unique referential index. Here we find one of the big payoffs of taking topological spaces seriously: we can define terms in a syntactic structure as locations in the workspace, and assign addresses to each syntactic term (as in Sarkar & Joshi, 1997; see also Gorn, 1967): these are unique identifiers for nodes in a graph, which can be related to locations in memory (see fn. 3) or -in this case- points in the workspace. Once we do that, we can call each address as many times as needed; that is, as many times as there are 'copies' of a syntactic term in distinct local neighbourhoods. If we do not pay attention to their phonological form (*John* vs. *him*), then we can simply assign each syntactic node an address that corresponds to the semantic value (or *interpretation*[11]) of that node. Let Δ stand for the address which points towards the interpretation of *John*: Δ specifies the location of the interpretation of *John* in the workspace. What this means is that syntactic objects in structural descriptions are *not tokens* of lexical items, but *types*. This implies a departure from (some versions of) MGG, since

> (. . .) *he* [Chomsky] *wants LA* [a Lexical Array] *to be not a set of lexical types, but rather a set of tokens.* [. . .] *Chomsky wants to identify chains at LF as equivalence relations over the objects in the numeration, but for that he needs lexical tokens, not types* (Uriagereka 2008:16. Our emphasis)

[10] The *order* imposed on a graph is both *total* and *strict* (which in turn implies that for any nodes *a*, *b* or any sub-graphs G, G', the order is *irreflexive*, *antisymmetric*, and *transitive*).

[11] This *interpretation* can (but need not) be defined, as in Krivochen (2018b: 31), as the translation of the node into intensional logic (in the sense of Montague, 1973).

This is an important point, since it highlights what a proper definition of workspaces can do for us. In the structural description of (18) there are two distinct objects *John* and *him* only if a structural description is a set of lexical tokens: the word 'John' and the word 'him' are lexical tokens introduced at distinct points in the derivation and bearing no relation to each other unless an index says so. However, because of the way in which workspaces have been defined here, structural descriptions are simply specifications of a lexical predicate, its arguments, and its functional modifiers (i.e., of a lexical anchor and its neighbourhood). The composition of neighbourhoods by means of substitution involves also the identification of all common nodes with the same address. Then, we need to revise (18) as in (19):

19) a. $[_{M}$ Δ wished [L]]

b. $[_{L}$ that Mary would date Δ]

c. $[_{M}$ Δ wished $[_{L}$ that Mary would date Δ]]

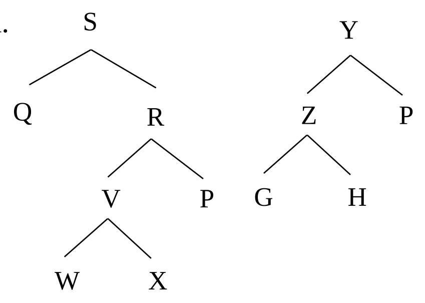
If syntactic nodes are addresses, then once L substitutes for L in M, we simply have *a single instance of the address* Δ *in two distinct syntactic contexts*[12] (where the *context* of a syntactic object A is the set of nodes A is immediately dominated by and the set of nodes that A immediately dominates). Crucially, once we define a walk through the derived structure (19c), Δ in M is ordered before Δ in L, since L is embedded in M. Note that the operation *substitution* itself (in MGG as well as TAG) is allowed because the grammar is capable of identifying identical labels; what we call L is also an address, which corresponds to the neighbourhood of *date*. The process of substitution and

[12] In Krivochen (2015b) these were considered *tokens* of the same *type*. The argument in that paper does not carry over automatically, since in that context there was still a distinction between Lexicon and syntax, a distinction that we dispense with here. The mechanism in play in that work was dubbed *token collapse*, and also assumed that tokens are points in a workspace (Krivochen, 2015b: 279):

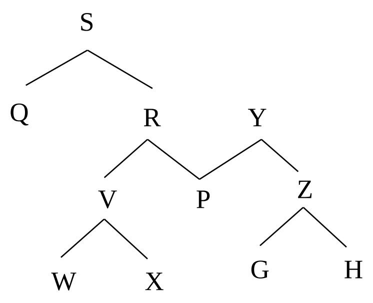
> *Token-collapse*
> *Let S be a set {a, b, . . ., n} of arbitrarily complex tokens in positions P within a derivational workspace W. An Interface Level IL will establish a dependency between the members of S iff:*
>
> *a. The members of S are mutually disconnected*
> *b. No two members of S belong to the same domain D, and there is no syntactic object SO, such that SO ∈ D and SO is logically equivalent to a member of S for interface purposes*
> *c. The members of S are structurally identical as far as format is concerned* [i.e., all terminals or all nonterminals]

adjunction depend on the grammar being able to read addresses, and addresses make sense if they point *somewhere*: that somewhere is a location in the workspace. These addresses serve to identify corresponding expressions: if nodes on distinct elementary trees T and T' are assigned the same address (say, P in (20) below), then a derived tree that contains T and T' will collapse those nodes into one (note that there is no requirement that the derived tree be single-rooted; see Morin & O'Malley, 1969):

20) a.



b.



Importantly, as long as the relevant node receives an interpretation in all elementary trees where it occurs, the neighbourhoods of two or more lexical predicates may share more than one node (see (25) below). This is precisely what we need for (18): the elementary structure (18a) contains an address (which we called L) that coincides with the root of the structure (18b) in its frontier; in addition to this, there is another common address embedded in both structures (which we called Δ).

Identity is not defined in terms of indices or external elements added to the representation (which also require the grammar to recognise identical indices), but simply in terms of where in the space the addresses point to. It is crucial to note that, if syntactic operations are required to yield tree-like structures in which an element cannot be dominated by more than a single node (the so-called Single Mother Condition) then we are *required* to multiply the entities in the structural description: because syntactic context is defined in terms of dominance, Δ dominated by the root in M and Δ dominated by *with* in L already have a mother node and cannot have another; in MGG each needs to be a distinct object for purposes of syntactic operations. Under strict set-theoretic assumptions, where M and L are sets of (sets of) syntactic terms, we cannot say that Δ in M and Δ in L (call them Δ_M and Δ_L) are the same object, since it would entail that Δ both belongs and does not belong to the set L.

However, this multiplication of entities does not arise in the present proposal; topologically, the identification of Δ_M and Δ_L amounts to having $d(\Delta_K, \Delta_L) = 0$. This is possible because we are defining graphs, not sets. A topological perspective on the notion of *workspace* allows us to simplify the mechanisms of the grammar, in this case dispensing with independent indexing mechanisms for identical terms in distinct syntactic contexts.

More interesting issues arise when we consider complex cases, such as (21a, b):

21) a. [Which picture of himself$_j$]$_i$ did John$_j$ say Mary likes __$_i$
b. John$_i$ wondered [which picture of himself$_{i/j}$]$_k$ Bill$_j$ saw __$_k$ (Chomsky, 1995: 205, ex. 36 a)

The approach to chain formation in Martin & Uriagereka (2014: 174, ff.) allows for a phrase marker to 'fold', collapsing distinct links of the chain into a single object. In their view,

Chains are best represented as being comprised of several simultaneous derivational stages, so that in principle they exist in one or the other stage (say, the 'foot' or the 'head' of the chain, in these instances). To interpret a chain in a particular chain-state $\boldsymbol{\rho}$ *is to collapse the chain in* $\boldsymbol{\rho}$.

Under present assumptions, there is no 'chain collapse' mechanism, because there are no chains *stricto sensu*: the workspace contains a set of local neighbourhoods of lexical heads (*elementary trees*), each of which is in turn a set of addresses and relations (e.g., the binary relation <). However, there are some similarities between Martin & Uriagereka's treatment of long-distance dependencies and our own. Let us consider what structural description of (21a) would look like, indicating only the variables that correspond to *John* (whose address will be denoted by Δ) and *Mary* (whose address will be denoted by Φ). A first approximation could be (22):

22) [Which picture of Δ] did Δ say Φ likes

But this cannot be right, because there is a 'gap' licensed by the transitive verb *like*. Regardless of how we represent filler-gap relations, there has to be a way to indicate that the term [which picture of Δ] satisfies the valency of *like*, but it also receives an operator interpretation:

23) For which *x*, *x* a picture of John, John said Mary likes *x*

In this case, *substitution* at the frontier would not work. The reason is that there is a Δ in the subject position of the matrix clause, and a Δ within the operator complex [which picture of Δ], there is no *root-frontier* relation (as opposed to the situation in (18)). Furthermore, this operator complex also appears in two contexts, as evidenced informally in (23). How would this be solved in a transformational, combinatory-based syntax? By multiplying the nodes and incorporating a notion of *indexing* that takes care of identification whenever relevant:

24) [Which picture of $\text{himself}_j]_i$ did John_j say Mary likes t_i

The derivation of (24) along MGG lines requires, at least, the following:

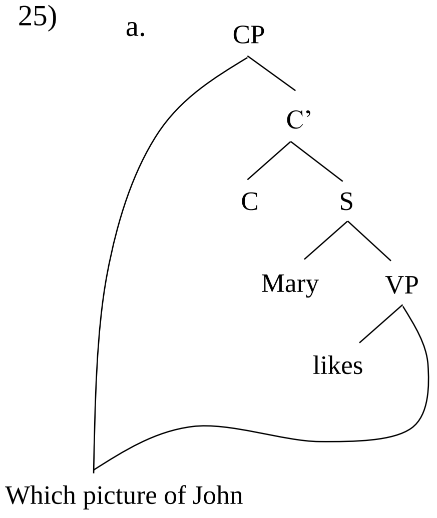
- A set of operations (phrase structure rules, EM, MERGE, etc.) to generate the string *John said Mary likes which picture of himself*
- A movement rule that displaces the syntactic term [which picture of himself] from its base position as the complement of *like* to the 'left periphery' (adjoined to the root), call it *Wh-movement*, IM, etc.
- An indexing rule that keeps track of occurrences of syntactic terms. It needs to be able to assign the same index to *John* and *himself*, but also to *which picture of himself* and *t*.

We will not deal in detail with these operations, in particular, *wh*-dependencies are outside the scope of the present paper (but see Krivochen, 2018b: §9). For purposes of this work, we can summarise what we want the system to capture:

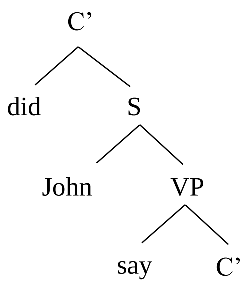
- *John* and *himself* denote the same entity (or, as in Krivochen, 2018b, they have the same intensional logic translation)
- The syntactic object *Which picture of himself* occurs in two syntactic contexts

Because we will not say much about it, let us begin with the latter point. In a TAG, this kind of dependency requires us to define an elementary tree where *which picture of John* appears in an A'-position linked to a gap in an A-position: non-local dependencies reduce to local dependencies (within a single elementary tree) once recursively introduced structure is factored out (Frank, 2002: 27). If we assign an address to the operator complex *which picture of John*, then we have the following simplified elementary trees (Frank, 2002: Chapter 5), each of which contains exactly one lexical predicate:

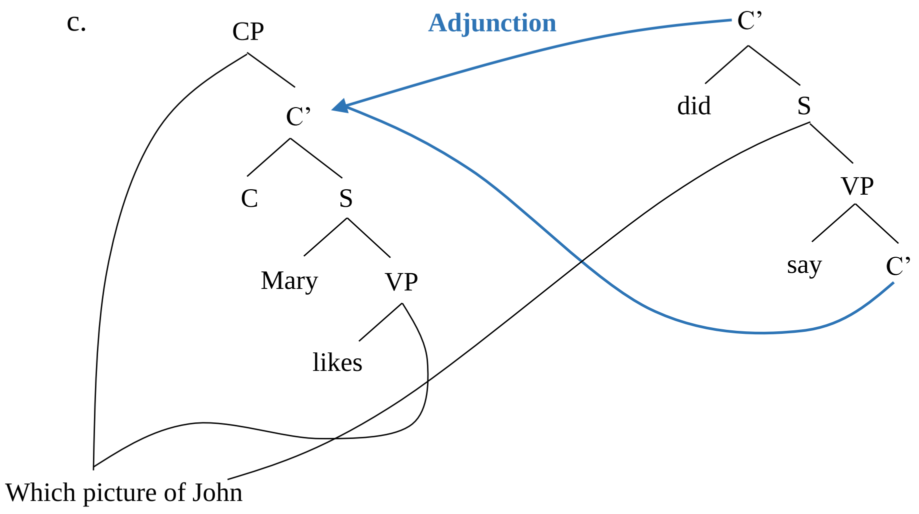
25) a.



b.

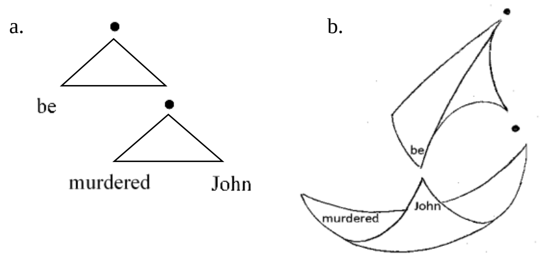


c.



A condition of TAGs is that links between displaced constituents and their 'base' position are preserved under adjunction (Joshi, 1985); in our terms, it means that the *distance* between an operator and a variable does not change if we adjoin an elementary tree 'between them'. Having *which picture of John* be assigned an address which is called from two distinct syntactic locations allows us to capture this basic TAG condition in terms of properties of a *metric* space: the distance between *which picture of John* and itself is always 0, regardless of which elementary tree we consider. MGG, by multiplying objects in terms of copies or traces, would assign a positive non-zero distance between the ‘copies’ of the operator complex. This does not mean that the distance between *other nodes* and *which picture of John* is 0, however: defining a walk through the derived tree from any other node to the operator complex will give us a non-zero distance. The address assigned to the operator complex must preserve information about the lexical category that it contains: in Gorn (1967) logical symbols (parentheses, punctuation, arrows, etc.) are not assigned addresses, only ‘object characters’ are; furthermore, the address of a non-terminal node is calculated based on the addresses of the nodes it dominates (because of the phenomena that he wanted to systematise, Gorn annotates edges rather than nodes; Sarkar & Joshi 1997, adapting Gorn’s system, assign addresses to nodes). In the operator complex *which picture of himself*, *which* and *of* would not be assigned an address: *which* introduces a choice function (Reinhart, 1998) and *of* relates two NPs but is not itself a lexical category. The address of the operator complex must capture this: the lexical category *John* must be accessible in that address: we can notate this as {picture, {John}} (which also indicates that *John* is an argument of *picture*); at this point in the development of the theory, notational issues are secondary. In a TAG, the elementary tree (25b) can be ‘shoehorned’ into (25a) at the node C’ (which is at both the root and the frontier of (25b)); this yields a derived tree (25c) where all nodes with the same address are ‘collapsed’ (i.e., reduced to one) since their distance is 0 (by the *identity property* of metric spaces). The core idea is that if we unify two (or more) elementary trees all of which contain an expression with the same address, the resulting structure will only contain *one* such expression, which enters into dominance relations with nodes in each of the elementary trees[13].

[13] The use of crossing lines is no more formally significant than the use of the colour blue for the arrows signifying adjunction. The relevant operation is the identification of common addresses in the workspace. This can be captured graph-theoretically, by defining the relations of dominance between nodes in elementary trees or in purely topological terms, by having the workspace fold onto itself. In this case, for the passive *John was murdered*, we would be mapping a space like (a) into a space like (b):



Let us now focus on the first point. We can express it in our terms as follows:

26) $d(John, himself) = 0$

But we know that (26) can only hold if *John = himself* by (12b), which is equivalent to saying that in (26) *John* and *himself* are addresses to the same location in the workspace[14]. But this is, obviously, not necessary. We can have an example like:

27) Which picture of John does John think Mary likes?

Where the *John* that thinks is not the same that appears in the picture (e.g., *John Paul Jones thinks Mary likes a picture of John Lennon*): in this case, we would say that there is a 'repetition' of the word 'John'. An interesting problem arises. Chomsky and others pose the following question: '*How does the interpretive system know what's a copy and what's a repetition?*'. As far as we can see, this is only a problem if the interpretive system has access to the the terminal string of the derivation and must define semantic interpretation as a function of that. Let us look at some simple cases:

28) a. John admires John
b. Himself, John admires
c. John admires himself

The two *John*'s in (28a) do not refer to the same entity: they are two distinct addresses. Thus, they would be *repetitions*. In (28b) there would be a *copy* of *himself* in the position of Compl-V and another in the left periphery. In (28c) whether there is a copy or not would depend on the reader's preferred theory of anaphora: if anaphoras are Spell-Out effects of having two co-indexed NPs in a very local environment (Lees & Klima, 1963; McCawley, 1970; Grohmann, 2003: Chapter 3;

In this view, pursued in Krivochen (2018a) and Saddy (2018), phrase markers are notations for manifolds in the workspace, and the topological transformations that can give rise to 'folded' configurations like (b) are homomorphic mappings: they preserve relations within selected structure. This contrasts with the idea that Merge (or MERGE) creates sets: as pointed out by Colling & Groat (2018), from {John, {was, {murdered, John}}} we cannot determine that both instances of the word 'John' correspond to the same syntactic object. An indexing system (and rules to determine index identity and interpretation) becomes inescapable not because of reasons of empirical adequacy, but because of the commitment to sets (see Collins, 2017).

[14] This is simply a way to capture the core insight of the identity condition for the application of a *pronominalisation* rule, assumed in Lees & Klima (1963) all the way through to Hornstein & Idsardi (2014). Of course, this view does not entail that *all pronouns* are the result of *pronominalisation*-like operations. Pronouns in root clauses need not be transformationally derived, as argued by Lakoff (1976: 329, ff), among others.

Hornstein & Idsardi, 2014, among many others)[15], then a copy is needed (cf. Grohmann's *Copy Spell-Out*). At this point, if we turn to Chomsky (2020), the scenario is quite unclear. Chomsky (2020: 45) says that '[we can define] *copy as just anything produced by MERGE (and no other rule)*'. This is problematic on multiple accounts. First, if MERGE subsumes both Merge and Move (as Chomsky suggests explicitly on p. 44), it is not clear what other rule is there in the grammar that can 'produce' things. The grammar presumably also contains a rule of Agree, but that rule does not introduce syntactic objects (it operates at a different level of granularity: that of features). Second, in MERGE understood as Merge (i.e., External Merge) there must be a *removal* operation in addition to a *copy* operation (again, see Chomsky, 2020: 44). But the explicit mechanisms by means of which syntactic objects are removed, and a specification of *where* they are removed from are not specified. From the workspace, presumably, but without a proper definition of *workspace* the whole exercise seems pointless. The third problem is that, if

> […] *MERGE always produces two copies, but in the case of external MERGE, just by the nature of the operation, the minimal operation, only one of them remains* (Chomsky, 2020: 44)

To account for (28b) in addition to MERGE creating two copies and leaving them both active we need an indexing mechanism: indexing tells us that *John* and *himself* denote the same entity, then MERGE creates a copy and introduces it back into the workspace above the root, without deleting or removing anything (deletion of the lower copy of *himself* would take place at PF).

In the present view, the definition of workspace must play a role in accounting for these facts. What becomes relevant in (28a) is that both instances of the word 'John' do not correspond to addresses to the same point in the workspace: they receive a different interpretation. This is given, in the sense that (28a) is not (cannot be) interpreted as (28c): we need to provide an adequate description of why this is the case. If the syntactic computation manipulates lexical tokens in strictly binary-branching, Single-Mother-Condition-abiding trees, then it is indeed difficult to see how we could

[15] In the present framework, 'very local' means 'within a single elementary tree'. Specifically, an elementary tree that is the neighbourhood of a transitive verb and which does not include a passive auxiliary.

differentiate between (28a) and (28c); however, if we consider the definition of *neighbourhood* that we have been using we can say something interesting. The crucial fact about (28c) is that the same entity is the subject and object of *admire*. This can be captured in structural descriptions like (29), where the directed nature of the graph is indicated with arrows:

29)

$$\begin{tikzcd}
& S \arrow[dl] \arrow[dr] & \\
NP & & VP \arrow[dl] \arrow[ll, bend left=70] \\
& V &
\end{tikzcd}$$

In (29a) the node NP is directly dominated by the root and is the sister of VP, which makes it the Subject of the sentence, but it is also dominated by VP and is the sister of V, which makes it the Object (Chomsky, 1965: 69). Note that the requirement of a total order is respected: dominance is total, antisymmetric, and transitive. The search sequence in (29), breadth-first, would be:

Σ = <S, NP, VP, V, NP>

In a configuration like (29), the second visit ('second' in terms of the strict order imposed on nodes in the graph) to N within an elementary tree corresponds to the Spell-Out of N as an anaphora; if the node address is called again from a *different* elementary tree, the Spell-Out would be pronominal. The crucial aspect of this is that the insight that a predicate is reflexive if two of its arguments (the external and an internal argument) are coindexed (Reuland & Reinhart, 1993) can be captured without the need to invoke indices: what we need is a condition like (25), which is given by the definition of the workspace as a metric space. In this context, the difference between (28a), which in MGG terms would involve *repetitions* (of a word) and (28c), which would involve *copies* (assuming a transformational origin for anaphoras) is not something that the 'interpretative system' has to invoke any special operations to account for: in one case there is one node that is visited more than once, in the other, there are two distinct nodes[16].

[16] Collins & Groat (2018) consider a multidominance approach (specifically along the lines of Citko, 2011, which has some differences with respect to the framework explored here; see also Johnson, 2016) to the

As a consequence of our definition of the workspace and what its points correspond to, we do not need to incorporate any additional terminal node or indexing mechanism to account for so-called *copies*. Let us go back to (24). In a lexicalised grammar where every lexical head projects an elementary tree (and not just lexical predicates), *picture* would also head its own elementary tree, which means that we need to consider, on the one hand, the interpretation of the address corresponding to the word 'John'; on the other, the interpretation of the elementary tree where that address occurs. We can present the procedure as follows:

30) 1. Define the neighbourhood of *picture* (this requires us to define the walk through the graph which corresponds to this neighbourhood)

2. Define the neighbourhood of *like*; this requires us to define the neighbourhood of *picture* as a proper subset in the neighbourhood of *like*

3. We have the interpretation of the neighbourhood of *picture*, use that as part of the input for the (directly compositional) interpretation of the neighbourhood of *like*

4. Define the neighbourhood of *say*; this contains an element whose address is the same as an address in the neighbourhood of *like*

5. Compose the elementary trees by *substitution*

6. Unify all nodes with the same address[17]

---

distinction between copies and repetitions in Minimalism. Their critique of the multidominance approach begins as follows:

> *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 by rejecting a graph-theoretic approach and considering problems with a set like {John_1 {T, {be, {seen, John_1}}}}, 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 we also made in Krivochen, 2015b). The system proposed here does not require diacritics: either two 'John's are addresses to the same location in the workspace (and thusor they are not. Our system is based on graph theory (see in particular Krivochen, 2018b) rather than set theory, but even in a strict set-theoretic Minimalist approach the issue seems to be how syntactic terminals are defined and what nodes in a phrase structure tree stand for.

[17] Relevantly, if syntactic objects in the workspace are assigned unique addresses, then the 'violations' of so-called *Determinacy* (Chomsky, 2019, 2020; Komachi et al., 2019) do not arise: in a configuration like (i) (which would correspond to Parallel Merge, one of the 'extensions' that MERGE would set out to eliminate)

i) {*a*, *c*}, {*b*, *c*}

These need not be interpreted as sequential steps; because there is no separation between lexicon and syntax, the operations glossed in (30) bring some points in the space closer together, defining the neighbourhood of lexical heads. Thus, steps (30.1), (30.2), and (30.4) may proceed in parallel.

We said that elementary trees are sets of addresses and relations. In graph-theoretic terms, these relations can be identified with annotated arcs which indicate the grammatical function that a certain node is assigned (as in Arc Pair Grammar), or by defining a total order relation between addresses which defines a unique walk through the graph (Krivochen, 2018b). Using the second method, we can define two elementary trees, corresponding to the two lexical predicates *say* and *like*. Using Ojeda's notation for dominance relations in graphs, such that ($a < b$) means that *a* precedes *b* on a walk through G (in other words, *a* dominates *b*); then, we can have the following slightly simplified description for the declarative version *John says Mary likes a picture of himself* (with Dependency Grammars, we assume that predicates always dominate their arguments; see Osborne et al. 2011 for a discussion about the relation between MGG and DG):

31) Elementary Tree 1: [John say [S]]
Elementary Tree 2: [$_S$ Mary likes a picture of John]
ET 1: ⟨(say < John); (say < like)⟩
ET 2: ⟨(like < Mary); (like < picture); (picture < John)⟩

Dominance is a two-place relation, thus, the descriptions of local graphs in terms of dominance are ordered sets of pairs of nodes: this is because (as we said before) the workspace is populated by a set of strictly ordered graphs, which constitute the neighbourhoods of lexical predicates. The two elementary structures have *John* as a common node; and the matrix verb *say* dominates the root of what is its complement, the clause [Mary likes which pictures of John], which is the verb *like* (thus, it

If *a*, *b*, and *c* are addresses, then there are *not* two copies of *c*, but one call to the same memory location in two distinct syntactic contexts (a single *copy*, presumably). If the elementary tree ET = {*a*, *c*} is then linked to the ET = {*b*, *c*} at {*c*}, the issue becomes defining which grammatical relation is satisfied by this operation (e.g., is *c* an argument of *a* or of *b* or of both? -think, e.g., of raising-to-object structures-). But it is possible, once we have a concrete example that follows the structural pattern described in (i) (not provided in the references cited in this note), to define a directed graph and a walk in that graph that visits *c* in two instances (thus, a *trail* in the graph; we can then say in Arc Pair Grammar parlance, that *c* is the 1 of *a* and a chômeur of *b*, for instance. See Krivochen, 2018b for extensive discussion). A similar argument is, as far as we can see, valid for the other cases illustrated in Komachi et al. (2019: 280-281).

transitively dominates every node that root dominates). When the two structures are combined, the result is a derived structure where *John* is dominated by two other nodes: *picture* and *say* and in which *John* dominated by *say* precedes *John* dominates (transitively) by *like*. This is possible precisely because the system defines graphs in the workspace, rather than building sets.

This process of identification follows from the characterisation of the workspace, and proves particularly useful to address the problem of distinguishing between *copies* and *repetitions*. The notion of 'occurrence' used in the context of Chomskyan Minimalism not clear. The terms 'occurrence', 'copies', 'repetitions' have been used in a transformational framework (Collins & Groat, 2018; Chomsky, 1995, 2019, 2020), but they correspond to intra-theoretical entities which depend on there being displacement transformations and an indexing mechanism over elements in a Numeration and in the derivation. Many of the problems identified in Collins & Groat (2018); Collins & Stabler (2016); Chomsky (2019) arise because derivations operate over sets of tokens lexical items and sets of sets of tokens of lexical items, with workspaces being defined in terms of these (Collins & Stabler, 2016: Definition 10). Chomsky (2013: 40) uses copies and repetitions as types of occurrences, without formally defining either. In (2000: 115) he says that '*an occurrence of α in K to be the full context of α in K.*', where K is a syntactic object. Presumably, the 'context' refers to the mother-daughter nodes of α. There is a confusion, we think, between phrase markers as sets, phrase markers as graphs, and diagrams of phrase markers (see Postal, 2010: 7; McCawley, 1998: 47-48 for useful discussion about this point) that makes defining these objects and relations in an unambiguous way difficult. Making explicit the mechanisms of each is necessary in order to properly compare the empirical adequacy of set theoretic and graph theoretic syntax.

The approach adopted here is in this sense diametrically opposite to the Minimalist one: we are concerned with workspaces as topological spaces: in these spaces, elementary trees are defined in graph-theoretic terms. In the present framework, we may consider a derived structure which results from the composition of two or more neighbourhoods: how does interpretation proceed? A possibility is that the interpretation of a structure is defined as a *walk* in that structure: if each elementary tree is strictly ordered in that derived structure, the same node may be visited more than once in a walk through the derived structure: in (28), for example, the node corresponding to the address for *John* is visited twice, coming from different nodes. Because we are dealing with two visits to the same node, we can capture the idea of *copies* without multiplying the nodes: this multiplication is unavoidable in set-theory based syntax. When neighbourhoods intersect via *substitution* or *adjunction*, identical addresses in initially distinct neighbourhoods are treated as a single element, since it is just an instruction to retrieve an interpretation (or access a portion of the lexical space; see e.g. Manzini & Savoia, 2011: Introduction for related discussion).

4. Conclusions

We can now summarise some aspects of the theoretical proposal as has been presented so far. Our starting point was the contrast between the lack of attention to the formal properties of *workspace* in MGG and its centrality in operations of structure building and mapping. We need to determine what exactly we want a *workspace* to do for us:

- Intuitively, points in a space can be close or far apart to different degrees: we can call the function that defines just how close or far apart points are the *metric* of the space. We want to capture that.
- We want to preserve *distinguishability*: two *distinct* points A and B in a space X can be arbitrarily near or far apart, but never have 0 distance. A and B can have 0 distance in X iff A = B.
- We want to be able to define the neighbourhood of a point in the space, and determine whether that neighbourhood is *accessible* to other points in the space or not
- We want the space to work for us: it must make the definition of syntactic relations more transparent or explicit, it must capture some property of grammatical entities that would be otherwise overlooked or mischaracterised

In order to address these issues, the introduction of some mathematical concepts was unavoidable, and we believe also welcomed. We proposed an explicit definition of *workspace* in topological terms, and related the mathematical properties of the syntactic workspace to the kinds of operations and relations that can hold between terms in that space. Syntactic operations create intersections between the neighbourhoods of lexical heads; these are elementary structures as in LTAG (for which we have provided empirical motivation from Spanish auxiliary chains) and may be composed by means of substitution or adjunction. The metric space which results from the composition of local neighbourhoods displays topological distinguishability as well as variable distances between points (Willard, 2004; Munkres, 2000), which is what we want for syntactic structures.

This paper constitutes an effort to explore the concept of *workspace*, which plays an important role in current MGG, but which is seldom defined or implemented in actual syntactic derivations. Conceiving of syntactic terms as addresses to points in a space allows us to dispense with inaudible additional structure (the multiplication of nodes in *chains*) for dependencies involving so-called *copies*. Furthermore, we eliminate the distinction between a lexical space and a syntactic space (there is no Lexicon separated from C_{HL}): there is only *one* workspace which contains a set of directed graphs; 'syntax' is 'something that happens to the (topology of the) lexicon'. Problems related to the multiplication of workspaces in copying operations (External Merge as *copy from the lexicon to the syntactic workspace* and Internal Merge as *copy from and to the syntactic workspace*) thus do not arise. In the end, what we have is a strictly ordered graph defined in a *metric* space: nodes in graphs in the workspace are addresses to intensional interpretations. Also, it is important to point out that the operations *do not extend* the space, if anything, the portion of the lexical space that we care about gets literally smaller as points are drawn closer together. The system sketched here is in principle compatible with Chomsky's (2019) desiderata for a theory of operations in the syntactic workspace.

5. References

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

Beim Graben, Peter & Sabrina Gerth. 2012. Geometric Representations for Minimalist Grammars. *Journal of Logic, Language, and Information* 21(4). 393-432.

Bošković, Željko. 2016. What is sent to spell-out is phases, not phasal complements. *Linguistica* 56, 25-56.

Bošković, Željko. 2020. On the Coordinate Structure Constraint, Across-The-Board movement, phases, and labelling. In Jeroen van Craenenbroeck, Cora Pots and Tanja Temmerman (eds.) *Recent Developments in Phase Theory*. Berlin: De Gruyter.

Bravo, Ana, Luis García Fernández & Diego Gabriel Krivochen (2015) On Auxiliary Chains: Auxiliaries at the Syntax-Semantics Interface. *Borealis*, 4(2). 71-101. http://dx.doi.org/10.7557/1.4.2.3612

Cardinaletti, Anna & Giuliana Giusti. 2001. 'Semi-lexical' motion verbs in Romance and Germanic. In Norbert Corver & Henk van Riemsdijk (eds.), *Semi-lexical Categories: The Function of Content Words and the Content of Function Words*. Berlin/New York: Mouton de Gruyter. 371–414.

Carstens, Vicky, Norbert Hornstein & T. Daniel Seely. 2015. Head-head relations in Problems of projection. *The Linguistic Review* 33(1). 67–86.

Chomsky, Noam. 1955. *The Logical Structure of Linguistic Theory*. Mimeographed, MIT. Available online at http://alpha-leonis.lids.mit.edu/wordpress/wp-content/uploads/2014/07/chomsky_LSLT55.pdf

Chomsky, Noam. 1965. *Aspects of the Theory of Syntax*. Cambridge, Mass.: MIT Press.

Chomsky, Noam. 1995. *The Minimalist Program*. Cambridge, Mass.: MIT Press.

Chomsky, Noam. 2000. Minimalist Inquiries: the framework. Minimalist Inquiries: The Framework. In Roger Martin, David Michaels and Juan Uriagereka (Eds.) *Step by Step – Essays in Minimalist Syntax in Honor of Howard Lasnik*. Cambridge, Mass.: MIT Press. 89-155.

Chomsky, Noam. 2001. Derivation by Phase. In Michael Kenstowicz (ed.) *Ken Hale: A Life in Language*. Cambridge, Mass.: MIT Press. 1-52.

Chomsky, Noam. 2008. On Phases. In Robert Freidin, Carlos Otero, and Maria Luisa Zubizarreta (eds.) *Foundational issues in linguistic theory*. Cambridge, Mass.: MIT Press. 133-166.

Chomsky, Noam. 2012. Foreword. In Angel Gallego (ed.) *Phases: Developing the Framework*. Berlin: Mouton de Gruyter. 1-7.

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. 2020. The UCLA lectures. https://ling.auf.net/lingbuzz/005485

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

Cinque, Guglielmo. 2004. ‘Restructuring’ and Functional Structure. In Guglielmo Cinque (ed.), *Restructuring and Functional Heads. The Cartography of Syntactic Structures*, Volume 4. Oxford: OUP. 132–192.

Citko, Barbara. 2011. Multidominance. In Cedric Boeckx (ed.), *Handbook of Linguistic Minimalism*. Oxford: OUP. 119-142.

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

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

Epstein, Samuel; Husatsugu Kitahara & T. Daniel Seely. 2015a. Derivation(s). In *Explorations in Maximizing Syntactic Minimization*. London: Routledge. 1-23.

Epstein, Samuel; Husatsugu Kitahara & T. Daniel Seely. 2015b. 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.

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

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

Gajić, Ljiljana. 2001. On ultrametric space. *Novi Sad J. Math*. 31(2). 69-71

Gorn, Saul. 1967. Handling the growth by definition of mechanical languages. *Proceedings of the April 18-20, 1967, spring joint computer conference*. New York: Association for Computing Machinery. 213–224.

Grohmann, Kleanthes. 2003. *Prolific domains: on the anti-locality of movement dependencies*. Amsterdam: John Bejnamins.

Hazewinkel, Michiel (ed.) 2001. Topological space. *Encyclopaedia of Mathematics*. URL: http://www.encyclopediaofmath.org/index.php?title=Topological_space&oldid=40046

Hornstein, Norbert & Jairo Nunes. 2008. Adjunction, Labeling, and Bare Phrase Structure. *Biolinguistics* 2(1): 57–86.

Hornstein, Norbert & William Idsardi. 2014. A Program for the Minimalist Program. In Peter Kosta, Steven L. Franks, Teodora Radeva-Bork & Lilia Schürcks (eds.), *Minimalism and Beyond: Radicalizing the interfaces*. Amsterdam: John Benjamins. 9-36.

Jackendoff, Ray. 2011. Alternative minimalist visions of language. In Robert D. Borsley and Kersti Börjars (eds.) *Nontransformational Syntax*. Oxford: Wiley-Blackwell. 268-296.

Jayseelan, K. A. 2017. Parallel Work Spaces in Syntax and the Inexistence of Internal Merge. In Gautam Sengupta, Shruti Sicar, Madhavi Gayathri Raman & Rahul Balusu (eds) *Perspectives on the architecture and the acquisition of syntax*. Singapore: Springer. 115-136.

Johnson, David & Paul M. Postal. 1980. *Arc Pair Grammar*. Princeton, NJ.: Princeton University Press.

Johnson, Kyle. 2016. Toward a Multidominant Theory of Movement. Lectures presented at ACTL, University College, June 2016. Available at https://people.umass.edu/kbj/homepage/Content/Multi_Movement.pdf

Joshi, Aravind K. 1985. Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? In David Dowty, Lauri Karttunen, and Arnold Zwicky (eds.) *Natural Language Parsing*. Cambridge, Mass.: CUP. 206-250.

Joshi, Aravind K. & Anthony Kroch. 1985. Linguistic significance of Tree Adjoining Grammar. Ms. University of Pennsylvania. ftp://babel.ling.upenn.edu/facpapers/tony_kroch/papers/relevance3.pdf

Joshi, Aravind K. & Yves Schabes. 1990. *Parsing with lexicalized Tree Adjoining Grammars*. Technical report. Department of Computer and Information Science School of Engineering and Applied Science, University of Pennsylvania.

Kaplansky, Irving. 1977. *Set theory and metric spaces*. Boston: Allyn and Bacon.

Kato, Takaomi; Hiroki Narita, Hironobu Kasai, Mihoko Zushi and Naoki Fukui. 2016. On the primitive operations of syntax. In Koji Fujita & Cedric Boeckx (eds.) *Advances in Biolinguistics. The human language faculty and its biological basis*. London: Routledge. 29-45.

Kayne, Richard. 1984. *Connectedness and Binary Branching*. Dordretch: Foris.

Kayne, Richard. 1994. *The Antisymmetry of Syntax*. Cambridge, Mass.: MIT Press.

Kayne, Richard. 2018. The place of linear order in the language faculty. Talk delivered at the University of Venice, June 16, 2018. Available online at: https://as.nyu.edu/content/dam/nyu-as/linguistics/documents/Kayne%200118%20Venice%20The%20Place%20of%20Linear%20Order%20in%20the%20Language%20Faculty.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.

Kracht, Marcus. 2001. Syntax in chains. *Linguistics and Philosophy* 24. 467–529.

Krivochen, Diego Gabriel. 2015a. On Phrase Structure Building and Labelling Algorithms: Towards a Non-Uniform Theory of Syntactic Structures. *The Linguistic Review* 32(3). 515-572.

Krivochen, Diego Gabriel. 2015b. 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 thesis, University of Reading.

Krivochen, Diego Gabriel. 2018b. *Syntax as graph theory*. Ms. University of Reading. https://ling.auf.net/lingbuzz/003842

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 & Susan F. Schmerling. 2020. A categorial grammar of Spanish auxiliary chains. Ms. Under review.

Kuroda, Sige-Yuki. 1976. A Topological Study of Phrase-Structure Languages. *Information and Control*, 30. 307-379.

Lakoff, George. 1976. Pronouns and Reference. In James D. McCawley (1976) (ed.). *Notes from the linguistic underground* (Syntax and Semantics 7). New York: Academic Press. 275-336.

Lees, Robert & Edward Klima. 1963. Rules for English Pronominalization. *Language*, 39(1). 17-28.

Manzini, Maria Rita & Leonardo Savoia. 2011. *Grammatical categories: Variation in Romance languages*. Cambridge: CUP.

Martin, Roger & Juan Uriagereka. 2014. Chains in Minimalism. In Peter Kosta, Steven L. Franks, Teodora Radeva-Bork and Lilia Schürcks (eds.), *Minimalism and Beyond: Radicalizing the interfaces*. Amsterdam: John Benjamins. 169-194.

May, Robert. 1985. *Logical Form: Its Structure and Derivation*. Cambridge, Mass.: MIT Press.

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

McCawley, James D. 1970. Where do Noun Phrases come from? In Roderick Jacobs & Peter Rosenbaum (eds.) *Readings in English Transformational Grammar*. Waltham: Ginn & Co. 166-183.

McCawley, James D. 1998. *The Syntactic Phenomena of English*. 2 Vols. Chicago: University of Chicago Press.

Morin, Yves-Charles & Michael O'Malley. 1969. Multi-rooted vines in semantic representation. In Robert Binnick et al. (eds.) *Papers from the Fifth Regional Meeting of the Chicago Linguistic Society*. University of Chicago. 178-185.

Munkres, James R. 2000. *Topology*. 2nd Edition. Prentice Hall.

Müller, Gereon. 2004. Phrase Impenetrability and Wh-Intervention. In Arthur Stepanov, Gisbert Fanselow, & Ralf Vogel (eds.), *Minimality Effects in Syntax*. Berlin: Mouton/de Gruyter. 289-325.

Nunes, Jairo. 2004. *Linearization of Chains and Sidewards Movement*. Cambridge, Mass.: MIT Press.

Ojeda, Almerindo. 1987. Discontinuity, Multidominance, and Unbounded Dependency in GPSG. In Geoffrey Huck & Almerindo Ojeda (eds.) *Syntax and Semantics 20: Discontinuous Constituency*. New York: Academic Press. 257-282.

Ouali, Hamid. 2010. Computation efficiency and feature inheritance in crash-proof syntax. In Michael T. Putnam (ed.) *Exploring Crash-Proof Grammars*. Amsterdam: John Benjamins. 15-30.

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

Prusinkiewicz, Przemyslaw & James Hanan. 1989. *Lindenmayer systems, fractals, and plants*. Berlin: Springer-Verlag.

Reid, Miles & Balász Szendröi. 2005. *Geometry and Topology.* Cambridge: CUP.

Reinhart, Tanya. 1998. Wh-in situ in the framework of the minimalist program. *Natural Language Semantics* 6. 29‑56.

Reuland, Eric & Tanya Reinhart. 1993. Reflexivity. *Linguistic Inquiry* 23(4). 657–720.

Richards, Marc. 2011. Deriving the Edge: What's in a Phase? *Syntax* 14(1). 74-95.

Rizzi, Luigi. 2016. Labeling, maximality and the head – phrase distinction. *The Linguistic Review* 33(1). 103-127.

Saddy, Douglas. 2018. Syntax and Uncertainty. In Angel Gallego & Roger Martin (eds.) *Language, Syntax, and the Natural Sciences*. Cambridge: CUP. 316-332.

Sarkar, Anoop & Aravind K. Joshi. 1997. Handling coordination in a tree adjoining grammar. Technical report, University of Pennsylvania. https://www2.cs.sfu.ca/~anoop/papers/pdf/tag-coordination.pdf

Seely, T. Daniel. 2006. Merge, Derivational C-command, and subcategorization in a label-free syntax. In Cedric Boecks (ed.), *Minimalist Essays*. Amsterdam: John Benjamins. 182-217.

Shieber, Stuart. 1986. *An Introduction to Unification-Based Approaches to Grammar*. Brookline, Mass.: Microtome Publishing.

Stroik, Tom. 2009. *Locality in minimalist syntax*. Cambridge, MA: MIT Press.

Stroik, Tom & Michael T. Putnam. 2013. *The Structural Design of Language*. Oxford: OUP.

Sutherland, Wilson. 2009. *Introduction to metric and topological spaces*. [2nd Edition]. Oxford: OUP.

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

Uriagereka, Juan. 2008. *Syntactic Anchors: On Semantic Restructuring*. Cambridge: CUP.

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

Willard, Stephen. 2004. *General Topology*. New York: Dover.

Zeijlstra, Hedde. 2020. Labeling, selection, and feature checking. In P. Smith, J. Mursell & K. Hartmann (eds.), *Agree to Agree: Agreement in the Minimalist Programme*. Berlin: Language Science Press. 137-174.

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