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 keeping 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 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. This paper is an attempt to (a) analyse the notion of 'workspace' as used in current Minimalist syntax, (b) recognise inconsistencies in the literature, and (c) provide a definition of 'syntactic workspace' that can help us capture interesting empirical phenomena. We explore the concept of *workspace* and its role in current generative theory, aiming at a mathematical characterisation of what workspaces are and how their properties determine possible syntactic configurations. In doing this, we will confront set-theoretic and graph-theoretic approaches to syntactic structure in terms of the operations that can affect syntactic terms. We analyse the consequences of conceptualising ‘syntax’ as a set of operations that affect local regions of spaces defining directed graphs, which correspond to elementary trees in a lexicalised tree-adjoining grammar.

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

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, 2020a, b; Chomsky et al., 2019; Komachi et al., 2019; Epstein et al., 2020), but a formal definition of workspace, a detailed discussion about what a workspace actually is, what adding a workspace to the grammar entails, 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 explore 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 departs from current generative practice insofar as it defines structural descriptions in *graph-theoretic* terms as opposed to the *set-theoretic* perspective that is standard in Minimalism. Because graph theory will be essential for this paper, we need to define what a *graph* is: a graph is a set G = (V, E), where V is a set of *vertices* and E is a set of *edges*. $v \in$ V is a vertex, and $e \in$ E is an edge. An edge e joining vertices a and b is notated $e = <a, b>$, and a and b are said to be *adjacent vertices*. If edges in a graph are 'one-way roads' connecting a *head* and a *tail*, they are referred to as *arcs*, and the graph is a *directed graph* (or *digraph*). The *neighbour set* of v is the set of adjacent vertices to v, usually notated N(v), and the *degree* of v is the number of *edges* connected to it (van Steen, 2010, Wilson, 1996).

Exponents of the graph-theoretic approach include Zwicky & Isard (1967), McCawley (1968), Morin & O'Miley (1969), Kuroda (1976), Johnson & Postal (1980) for Arc Pair Grammar, Postal (2010) for Metagraph grammar, Gärtner (2002), among others. In these works, syntactic operations establish dependencies between nodes and generate formal objects known as *topological graphs* (representations of graphs in a plane). Again, the graph-theoretic inclinations of early generative grammar contrasts with the Minimalist custom of defining syntactic objects and structure building operations in set-theoretic terms (Chomsky, 2008, 2019, 2020a, b; Epstein et al., 2015; Collins, 2017)[1]. Bearing this difference in mind is crucial, for some objects, relations, and operations that can be defined in graph-theoretic terms become impossible to define in set-theoretic terms (e.g., search algorithms cannot be defined for unordered sets, but they can for directed graphs; see Branan & Erlewine, 2021; Krivochen, 2021 for discussion). In this work, the distinction between set-theoretic and graph-theoretic approaches will become crucial for the definition of workspaces.

[1] Postal (2010: 8), for example, says that '*Unfortunately, most of the development of* [MGG] *theorizing has taken place with little or no exploitation of, or even serious recognition of, links to graph theory*'. See also Postal (2010: 392, fn. 3).

Above we mentioned long-distance dependencies as a phenomenon that requires ‘holding on’ to structure. Let us see an abstract example, from a Minimalist (set-theoretic) perspective. Assume that we have objects X and Y which are in a local relation, as represented in (2):

2) {X, Y}

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) {W, {X, Y}}

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 under set-theoretic assumptions (recall that Merge generates unordered sets) can either (a) move Y to a higher position in the checking domain of W, 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) a. {Y_i, {W, {X, t_i}}}

b. {Y, {W, {X, Y}}}

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 the structure; in (4b) we add a copy of Y and similarly expand the structure. In both cases, there is a local relation between W and Y, as required (because there is no other head between these two terms), but at the cost of introducing additional terms (a trace or a copy).

Moreover, the very idea of *copying* requires not only an operation that takes Y and somehow produces another Y (multiplying the entities), 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*. In Minimalism, 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).

1.1 Workspaces in the Minimalist literature: questions and problems

The generative procedure in MGG takes the form of an operation of 'simple binary set formation' (Chomsky, 2019; 2020a: 22; 2020b; see also Epstein et al, 2015a, b; Collins, 2017; Kitahara, 2020): Merge. Recently, Merge has been recast as MERGE, giving workspaces a central role. To give an 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.*

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

> *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'*[2].

[2] It is relevant to note that Chomsky does not say MERGE(P, Q), P ∈ W ∧ Q ∈ W; but rather '*MERGE applies to P and Q* ***and the workspace***' (Chomsky, 2020b; highlighting ours). If syntactic operations affect syntactic objects, then the workspace must be a syntactic object. If the workspace is not a syntactic object, then syntactic operations may affect things that are not syntactic objects. This poses questions and problems, as these operations must be defined in a non-standard way.

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[3]? 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, 2020a: 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, formally. 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. 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 accessible[4] 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*

[3] 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 (2020a: 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). Chomsky (2020b) defines the workspace as '*the set of objects that are available for further computation* […] *the workspace specifies the current state of computation*'; in this sense, the *workspace* is analogous to Turing's (1936: 231) definition of *configuration*. If External Merge involves access to the lexicon, then elements in the lexicon must be available for further computation (since there is no Select operation, and therefore no Lexical Array), thus belonging to the workspace.
[4] In Chomsky's recent works (2020a, b) *phases* are maintained as entities in the theory but defined in terms of designated 'endmarkers' for structural probing, *v** and C; crucially, in that approach locality is *not* linked to (or defined in terms of) properties of the workspace.

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[5]). It is still unclear whether the workspace is, to give just some 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. Part of this confusion arises from the lack of explicitness about whether a set of syntactic objects is always a syntactic object. If we look at the Minimalist literature, Collins & Stabler (2016: Definition 7) explicitly say that a set of syntactic objects is a syntactic object, and therefore a workspace is a syntactic object (Op. Cit.: 47). But not everybody agrees. 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 in the most recent version (Chomsky, 2019, 2020a, b) in the sense of 'a formal object that can be (part of) the input for rules of the grammar'. These issues are fundamental, but remain unaddressed in the literature, even the most theoretically oriented works. 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, 2020a), then how does it help at 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

[5] 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).

comprises the lexicon)? In sum: what exactly does proposing the existence of a *workspace* buy us in theoretical and empirical terms?

Let us briefly 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 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 itself 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, 2020a, b), new questions arise. Is the workspace co-extensive with a phrase marker or does it properly contain the phrase marker (if the latter, what else is there in the workspace?)? 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, 2015: 266)?

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 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, 2020a: 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; 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 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' structure. Interface levels and mapping rules become hard to avoid in such an architecture. 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 the Minimalist take on locality conditions: *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. In this paper we 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 and the objects generated by these operations: graphs, not sets (noting, furthermore, that these are not equivalent formalisations and that cannot always be mutually translated; cf. Chomsky, 1995: 226). 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 a set of discrete syntactic objects (features, lexical items, roots and categorisers…)[6].

2. Some properties of topological spaces

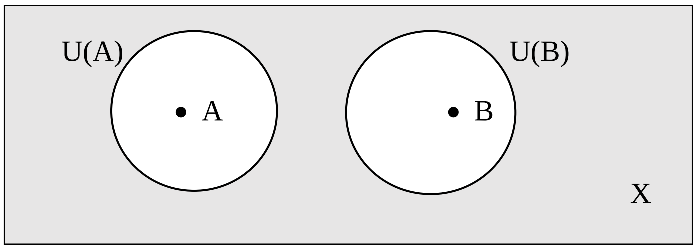
If we want the notion of workspace to work for us, theoretically and empirically, we need to provide an explicit characterisation of what a workspace is, what properties it has, and how it determines the kinds of operations that can apply in it and the format of objects that can serve as inputs and outputs 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. This approach implies a departure from the literature, as far as we can see, but is justified in that, as we will argue, it solves or avoids the problems and ambiguities that we identified in the previous section. First, then, we need to introduce the concept of *topological space* (see Sutherland, 2009: Chapter 5 for basic notational and

[6] 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.

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 other words, 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[7]. We can provide a graphical representation of disjoint closed neighbourhoods for A and B in a topological space X:

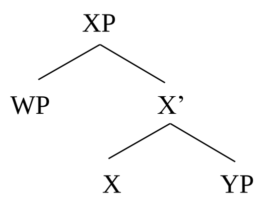
5)



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

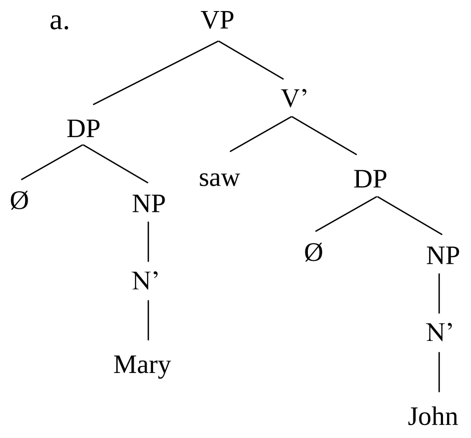
[7] We can choose between different axioms which relate points and their neighbourhoods, and the specific axioms that we choose gives us a classification of spaces. Furthermore, we can define functions that take us from one kind of space to another: the possibility to transition from one kind of space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

6)

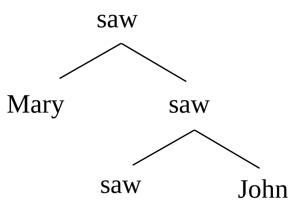


In X-bar syntax, we would say that X 'projects' an XP, an object whose distribution and semantic interpretation are determined by the properties of X, the 'head' of the phrase. As has been observed repeatedly in the literature (see in particular Chomsky, 1994; Collins, 2002; Seely, 2006; Gärtner, 2002; Hornstein & Nunes, 2008) there are problematic issues 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 Merge in a workspace (we will return below to the crucial issue of whether the diagrams in (7a-b) are just proxies for sets or should be interpreted in all seriousness as graphs; the choice between sets and graphs has major theoretical and empirical consequences). However, without a clear definition of what a workspace is, we cannot ask how the properties of workspaces are related to the properties of the operations that can apply and of 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 (specifically, its *c-projection*) 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). Graph-theoretically, as we saw, the neighbourhood of a node is the set of nodes directly connected to it: a head, in this view, must be directly connected to all its dependant nodes (this entails a move towards Dependency Grammars, see Osborne et al., 2011 for extensive discussion about how bare phrase structure and label-less trees indeed takes Minimalism in the direction of DG). This allows for an elimination of intermediate projections without additional stipulations.

Let WS be our workspace, with X, Y, and W points in WS. Then, the phrase 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 for probing 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 (adopted by different theories): lexical items, features, constructions / signs, roots and functional morphemes, or basic expressions. 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.

We are now in the position of providing some clarification about the identity of points in the workspace: we follow Sarkar & Joshi (1997) and Gorn (1967) in assigning each point in the space an *address* which corresponds uniquely to a region of the space[8]. This means that when a node appears in a complex object, that node is assigned a unique identifier that is independent of the syntactic context in which that node appears. As we will see in **Section 3**, the addresses assigned to nodes correspond to the *semantic values* of expressions (in the sense of Dowty et al., 1981; Heim & Kratzer, 1998). There is a single address for each elementary semantic value and a elementary single semantic value for each address: the definition of the addresses of complex objects, to which correspond derived semantic values, are obtained by means of composition operations (e.g., generalised transformations; Chomsky, 1955, 1995; Gärtner, 2002). The second substantive commitment that we make is that structures in the workspace are defined in graph-theoretic terms. This allows us to formulate operations to yield derived structures from local neighbourhoods in a straightforward manner: if nodes v_1 and v_2 on distinct neighbourhoods T and T' (such that $v_1 \in$ T and $v_2 \in$ T') are assigned the same address, then the composition of T and T' (call it T") will collapse those nodes into one, call it v_3: $v_3 \in$ T", for $v_1 = v_2 = v_3$ (Karttunen & Kay, 1985 use the term *structure sharing* for a process very similar to this one, but restricted to *binary trees* representing feature structures; this term has also been used extensively in HPSG, see e.g. Pollard & Sag, 1994: 19). Formally, the process is graph-theoretic *union* (but not *disjoint union*): let V_1 and E_1 be the set of nodes and edges, respectively, in a graph G, and V_2 and E_2 the set of nodes and edges in G'. Then, G ∪ G' = $(V_1 \cup V_2, E_1 \cup E_2)$ (Van Steen, 2010: 29; Wilson, 1996: 10). Our perspective is very close to Sarkar & Joshi's, in the sense that nodes in a graph are assigned memory addresses that can be called upon at different points in the interpretation of a structure (such that '*corresponding objects in the corresponding expressions have the 'same' addresses*'; Gorn, 1967: 214). The idea that points in the workspace correspond to uniquely identifying addresses has important consequences for the analysis of chain dependencies and binding, as we will see below.

[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 Manzini & Savoia (2011) for related discussion.

The definition of the 'projection' of a head as the *neighbourhood* of that head defined in the workspace allows us to clarify several issues: in particular, we can shed some light on the concept of *edge*. 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 been 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, are seen as parasitic on Agree (Chomsky, 2013, 2015; Zeijlstra, 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 (still assumed in Chomsky, 2020a, b) 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).

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 in workspaces as defined above 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, 2020a: 42, 56; accessibility is still restricted by additional principles 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, 2020a, b), 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, 2020a: 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 prevent accessibility[9].

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

Let us make an empirical stop at this point, to provide some empirical basis for the discussion about the relevance of *neighbourhoods* as defined above. 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' in order to define local domains, and cyclicity is driven by the opacity effects defined by the PIC. However, since phases are defined purely in terms of syntactic properties (phase heads bearing features that drive movement operations), 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). An alternative is to define syntactic cycles to be structured as the neighbourhood of *lexical* heads (as opposed to being the complement of functional heads, as in phase theory); this lexical head is referred to as the *anchor* of the local structure. If we do that, 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 building blocks of grammar are structures with a single lexical anchor is a *lexicalised* grammar (Joshi & Schabes, 1990; Sarkar & Joshi, 1997): being able to define the neighbourhoods of lexical heads in the syntactic workspace makes lexicalised grammars a natural and empirically fruitful 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):

[9] Having disjoint closed neighbourhoods for *x* and *y* entails that *x* is not accessible to *y* and *y* is not accessible to *x*, furthermore, no point in the neighbourhood of *x* is accessible to any point in the neighbourhood of *y* and vice versa. In syntactic terms, we may think of completely opaque domains: a syntactic term all of whose terms are inaccessible to operations triggered from outside that term, including substitution or adjunction. It is not clear whether such terms exist in natural language, but this is an empirical question. Much research pending, plausible candidates for units with *closed* neighbourhoods are syntactic objects that cannot be embedded (and thus not even their root can be targeted by syntactic rules): imperatives (Schmerling, 1982), vocatives, and interjections (Chomsky, 2008: 139).

Condition on Elementary Tree Minimality (CTEM)*: Each elementary tree consists of the extended projection of a single lexical head* (Frank, 1992: 53; see also 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. Krivochen & García Fernández (2019) 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)

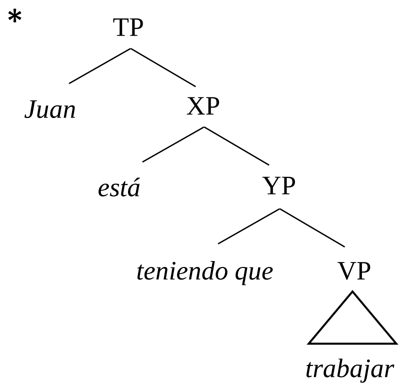
The definition of workspace proposed here allows us to capture the insights of LTAGs: elementary trees can be defined in terms of the workspace as the neighbourhoods of lexical heads. A major empirical advantage of LTAGs is that, because they are not committed to uniformly monotonic antisymmetric phrase 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, which differ greatly from the more studied English sequences of auxiliaries. 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'

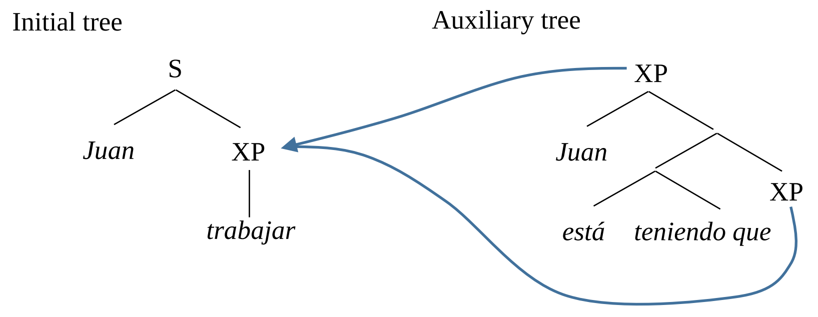
b. Juan está teniendo que trabajar
J. Aux.prog Aux.Mod.deont work
'J. is currently having to work'

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. Bravo et al. (2015) propose that this is so because *estar* does not have scope over *trabajar* in (9b), which in phrase structural terms means that *estar* cannot c-command *trabajar* (see e.g. 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 or any other template-based approach): both are equally 'basic'. From a syntactic perspective, the segmentation of a string like (9b) would similarly be problematic. Let us see why. Assigning a structural description like (9b') to (9b) is 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, cannot be satisfied by a monotonic clausal spine, but can be satisfied under TAG assumptions. Concretely, 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, for (9b) we need two elementary structures, simplified along the lines of (10) below. 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; Frank, 2013). In this way, the node in the initial tree can be replaced by the adjoined structure, as schematised in (10):

10)


As said before, labelled PS-style 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*. This is possible because, as argued in Bravo et al. (2015), modal auxiliaries are semi-lexical categories. 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*, which were at the core of Chomsky's 1995 formulation of Merge), 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 which are assigned uniquely identifying addresses 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. This contrasts with 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}} (see Jackendoff, 2011) *as per* the *No Tampering Condition* (Chomsky, 2008): neither set involved in Merge can be internally modified. A graph theory-based lexicalised grammar not only can define local derivational units in terms of neighbourhoods of lexical predicates in the workspace: it can also identify all common node addresses between the neighbourhoods of distinct lexical predicates and unify them into a single node (cf. Karttunen & Kay's 1985 *structure sharing*). In order to do this, however, the condition that a node may have only one mother must be ditched, thus yielding a *directed* graph with *multidominance* (see also Citko, 2011; Johnson, 2016)[10]. The conditions over the definition of local syntactic units and the way in which they are composed makes explicit reference to properties of the workspace.

[10] 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".*

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. 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* and *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)

‘Neighbourhoods’ as defined in the context of natural language syntax, are not only formal topological objects: they are objects subjected to requirements of semantic interpretation and externalisation. If we define neighbourhoods of points in the workspace in terms of the presence of (8a-c) above (such that the neighbourhoods of lexical predicates are what we call *elementary trees*), 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 non-monotonic derivations which get chunked into smaller domains which cannot be defined *a priori* (because 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). The definition of ‘syntactic workspace’ as a *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.

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. Here we defend the hypothesis that syntactic workspaces are topological spaces as defined above, with the additional consideration that some points are lexical heads (predicates, arguments), and others are functional heads (always modifiers). Now we need to examine some properties of spaces in terms of the axioms that define the possible relations between points. Different sets of 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):

11) For x, y, z points in a topological space S, S is ***metric*** iff:

a. $d(x, y) > 0$ if $x \neq y$ (*positive property*)

b. $d(x, y) = 0$ *iff* $x = y$ (*identity property*)

c. $d(x, y) = d(y, x)$ (*symmetric property*)

d. $d(x, z) \leq d(x, y) + d(y, z)$ (*triangle inequality*)

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*)

b. $d(x, y) = 0$ *iff* $x = y$ (*identity property*)

c. $d(x, y) = d(y, x)$ (*symmetric property*)

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. 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 $m < n$). Syntactic *distance* is measured in a number of ways (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: if we have a formalism for the workspace that allows us to capture aspects of *distance* without requiring additional mechanisms (e.g., an operation to count members of a set; Kitahara, 2020), it is a theoretical advantage.

A crucial property of metric spaces is that distances *sum*, as per the triangle inequality. The fact that distances between points vary and sum distinguishes metric spaces from *ultrametric* spaces. Gajić (2001: 96) provides a simple characterisation of ultrametric spaces:

Remark: *Let* [a topological space] $X \neq \emptyset$, [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 of defining a notion of workspace that is syntactically usable. In other words, we want a space that *preserves topological distinguishability* and in which variable distances can be defined unambiguously: locality conditions in the establishment of dependencies (either Minimality-based or impenetrability-based) are based on the idea that some syntactic objects are closer than others.

We have introduced ultrametricity because (a) we need to evaluate some alternative to metric spaces in order to properly argue for their adequacy, and (b) ultrametricity has a prominent role in some proposals about the mathematical nature of phrase markers in generative grammar that also adopt a topological perspective (Roberts, 2015; Uriagereka, 2012: 43-45). After all, ultrametricity has been used as a model for taxonomy and hierarchical structure in a variety of disciplines, including physics, data science, and biology (Murtagh, 2004; Rammal et al., 1986; Gavryushkin & Drummond, 2016). Furthermore, Hughes (2003) presents an explicit equivalence between infinite single-rooted trees (including Cantor and Fibonacci trees) and finite ultrametric spaces (see also Dovgoshey & Petrov, 2019); this is relevant since models of syntactic structure that propose a Fibonacci provenance for X-bar theory are forced to expand the X-bar schema infinitely (e.g., Uriagereka, 2014; Medeiros & Piattelli-Palmarini, 2018).

Here we depart from such proposals, and argue 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: as mentioned before, conditions like

Relativised Minimality (Rizzi, 2016), the Superiority Condition (Chomsky, 1977), Minimal Link/Shortest Move, Attract Closest (Chomsky, 1995) and others are, crucially, based on the idea that terms in a structural description can be at variable distances from one another. We will come back to this shortly. In this context, *metric* spaces become the natural option as a characterisation of syntactic workspaces: a workspace is a *metric topological space* whose points are *basic expressions of the language*.

The properties of ultrametric spaces, while not suitable 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 between nodes in the context of a phrase marker: this is how elementary trees are built. Following this reasoning, if the lexicon needs to maintain distinguishability but not have pre-encoded biases, then it is possible to characterise the lexicon as a space 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. Saddy argues that given the ultrametric inequality it is not possible to define notions like 'edge' in ultrametric trees, since the metric over the space is constant for any two distinct elements. 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: syntactic operations apply to the lexical space (see also Stroik & Putnam, 2013: 54). Structure emerges when the ultrametricity of the lexicon is disrupted by syntactic operations: merging X and Y entails bringing X and Y closer together than they were (Uriagereka, p.c.)[11]. Now the metric cannot be just two-valued: syntactic composition of X and Y puts X closer to

[11] In Uriagereka's terms, (p.c.)

> *The idea is that when you merge, say, "men" into "like arguments" (or some such),* ***you are literally getting "men" to a proximity w.r.t. "arguments" that it would not otherwise have had*** (as compared to, *say, "men" and "boys" or "arguments" and "discussions", say). As a consequence of the merge, each*

Y than to any other point in the space. Properties of the space where a phrase marker is defined, then, gives us some constraints with respect to the properties that can be ascribed to such construct. When the ultrametric space is metricised, we can define open neighbourhoods of lexical heads: these are elementary trees, the building blocks of syntax.

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 X, Y, and Z, whether a 'minimal' operation (e.g., MERGE, Simplest Merge, Agree, etc.) can establish a dependency of some sort (labelling, probing, theta-marking, etc.) between X and Y or X and Z, or whether Y and Z are equidistant from X. 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 (which allows us to define the *distance* between syntactic objects in the workspace) becomes an important part of an adequate meta-theory for these syntactic operations.

On the issue of accessibility, let us go back to the PIC_1 and PIC_2 above. We can compare the probing space that each of them allows for, by considering an abstract structure like (12):

12) $[_{\mathrm{CP}}\ldots$ [C $[_{\mathrm{TP}}\ldots$ [T $[_{v\mathrm{P}}$ Ext Arg [v $[_{\mathrm{VP}}\ldots$ [V Int Arg]]]]]]]]

In (12), 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.,

of the relevant words (understood as information-density peaks within the space) will obtain new conditions.

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 work with a notion of *distance* without making reference to the workspace. This is the road taken by recent research on labelling, based on Chomsky's (2013, 2015, 2020a) suggestion of a Minimal Search algorithm, which given a syntactic object finds a head (in Chomsky, 2020a: 48, this needs to be specifically the head of a chain) with suitable features for identification at the interface levels. But *distance* itself is not defined in the context of Minimal Search. Similarly, in those instances where MGG needs to resort to *equidistance* for purposes of feature-checking (Lasnik, 2009; Hornstein 2009: 42, ff.; Boeckx, 2008: 145), there is usually no formal definition of *distance*[12]. There are, however, some explicit proposals: for example, Hornstein (2009: 38) proposes that *distance* is based on comparing the set of nodes involved in a movement path for possible targets, with shorter paths being properly contained in longer paths (see also Kitahara, 2020). This set-theoretic approach to path comparison requires the grammar to be able to determine the cardinality of a set, such that it can compare it with the cardinality of another set to determine which path (*qua* set of nodes) is 'shorter'.

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

[12] For example, in Chomsky (1995) *equidistance* does not allow us to get to a working notion of *distance*:

> *The minimal domain Min(δ(CH)) of CH is the smallest subset K of δ(CH) such that for any γ ∈ δ(CH), some β ∈ K reflexively dominates γ.* (Chomsky, 1995: 274)
> *If α, β are in the same minimal domain, they are equidistant from γ.*
> *In particular, two targets of movement are equidistant if they are in the same minimal domain.* (Chomsky, 1995: 169)

elementary structures (the neighbourhoods of lexical predicates) and hinted at the existence of composition operations (substitution and adjunction) but we have not operationalised them.

3. Syntax in the workspace

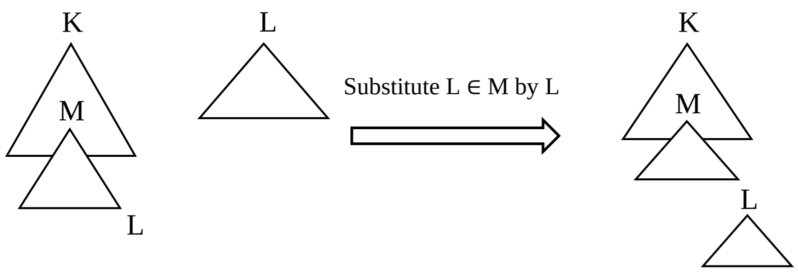
We have a characterisation of the workspace where syntactic operations apply (a metric topological space) and, in that space, we have the elements that appear in structural descriptions defined as points or sets thereof within said space. This metric space is populated by a set of *topological graphs*, which we called 'elementary graphs', defined as the minimal unit that contains a lexical predicate, its functional modifiers, and its arguments: each vertex (or 'node') in an elementary graph is a basic expression of the language assigned a uniquely identifying address. Elementary graphs 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, 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 'sub-components' be linked? The problem, when asked specifically about linguistic structures, pertains to the relation between locality in syntax and compositionality in semantics: within 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. Let us start with *substitution*. 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 L:

13)



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:

14) John wished [that Mary would date him]

(14) contains two clausal domains in a hypotactic relation; each of these corresponds to an elementary tree by virtue of containing a single lexical predicate (more specifically, being the neighbourhood of a single lexical predicate in the workspace). 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* (see also 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 (14) using *substitution* goes along the following lines:

15) 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 (15a) points to the same object as the address L in (15b): the root node of the sub-graph that is a structural description for the string *that Mary would date John*. Defining a parse of these graphs 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. However, this is not formally necessary. Whereas MGG defines *paths* in structural descriptions (Kayne, 1984: 132; 1994), theories that allow for multidominance (Gärtner, 2002; Citko, 2011) 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; this makes available structures that are impossible under set-theoretic assumptions. 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). Incidentally, this allows us to define *distance* in terms of walks: the distance between v_1 and v_2 in a totally ordered graph G is the number of nodes visited in a walk from v_1 to v_2. 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[13], and a strict order is a transitive relation, the derived structure (15c) is also strictly ordered. 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 and defining directed graphs on them: we can define terms in a syntactic structure as locations in the workspace, and assign addresses to each syntactic term: these are unique identifiers for nodes in a graph (i.e., 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 of that node. Let ⦃John⦄ stand for the uniquely identifying address of the expression *John*. 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)

This is an important point, since it highlights what a proper definition of workspaces can do for us. In the structural description of (14) 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 the formal relations defined in a directed graph containing a single lexical predicate, its arguments, and its functional modifiers (i.e., of a lexical anchor and its neighbourhood).

[13] 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*).

The composition of neighbourhoods by means of substitution also involves the identification of all common nodes with the same address (Sarkar & Joshi, 1997; Karttunen & Kay, 1985). Then, we need to revise (15) as in (16), where we substitute the expression *John* with the address ⦃John⦄:

16) a. [$_{\text{M}}$ ⦃John⦄ wished [L]]

b. [$_{\text{L}}$ that Mary would date ⦃John⦄]

c. [$_{\text{M}}$ ⦃John⦄ wished [$_{\text{L}}$ that Mary would date ⦃John⦄]]

Once L substitutes for L in M, we simply have *a single instance of the address* Δ *in two distinct syntactic contexts* (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 (16c), the node assigned the address ⦃John⦄ in M is ordered before the node assigned the address ⦃John⦄ 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; no additional mechanism is required in our framework since if nodes on distinct elementary trees T and T' are assigned the same address, then a derived graph that contains T and T' will collapse those nodes into one (see Sarkar & Joshi, 1997; Karttunen & Kay, 1985).

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 (22) below). This is precisely what we need for (15): the elementary structure (15a) contains an address (which we called L) that coincides with the root of the structure (15b) in its frontier; in addition to this, there is another common address embedded in both structures, ⦃John⦄.

Identity is defined in terms of the uniquely identifying addresses assigned to each basic expression of the language: expressions in a string are *identical* if and only if they correspond to a single node in a graph (basic or derived). 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 *date* in L already have a mother node each 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 ⦃John⦄ in M and ⦃John⦄ in L (call them $⦃\text{John}⦄_M$ and $⦃\text{John}⦄_L$ for convenience) are the same object, since it would entail that ⦃John⦄ both belongs and does not belong to the set L (see fn. 14, 18). However, this multiplication of entities does not arise in the present proposal as a consequence of the definition of the workspace: the identification of $⦃\text{John}⦄_M$ and $⦃\text{John}⦄_L$ amounts to having $d(⦃\text{John}⦄_M, ⦃\text{John}⦄_L) = 0$. This is possible because we are defining graphs, not sets, in the workspace: an account of the property of displacement in graph-theoretic terms does not need to make filler and gap distinct symbols (an idea that can be traced back to early Multidominance theories, e.g. McCawley's 1982 approach to Right Node Raising); it can work with *one* node visited more than once in a walk through an oriented graph. A topological perspective on the notion of *workspace* thus allows us to simplify the mechanisms of the grammar, in this case dispensing with independent indexing mechanisms for identical terms in distinct syntactic contexts (the notion of *chains* in displacement) in favour of a property of the space: the *identity property* (11b).

More interesting issues arise when we consider complex cases, such as (17a, b):

17) 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 (17a) would look like, indicating only the addresses corresponding to the expressions *John* (which we notated ⦃John⦄) and *Mary* (which we notate ⦃Mary⦄). A first approximation could be (18):

18) [Which picture of ⦃John⦄] did ⦃John⦄ say ⦃Mary⦄ 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 ⦃John⦄] satisfies the valency of *like*, but it also receives an operator interpretation:

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

In this case, *substitution* at the frontier (i.e., root-frontier node identification) would not work. The reason is that there is a ⦃John⦄ in the subject position of the matrix clause, and a ⦃John⦄ within the operator complex [which picture of ⦃John⦄], there is no *root-frontier* relation (as opposed to the situation in (15)). Furthermore, this operator complex also appears in two contexts, as evidenced informally in (19). 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:

20) [Which picture of himself$_j$]$_i$ did John$_j$ say Mary likes t_i

The derivation of (20) along MGG lines requires, at least, the following:

- A set of operations (phrase structure rules, EM, MERGE, etc.) to generate the terminal string *John said Mary likes which picture of himself*; in terms of the format of the structural description for that string, it is inevitably going to be strictly set-theoretical (due to *a priori* assumptions in the definition of Merge/MERGE)
- 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.

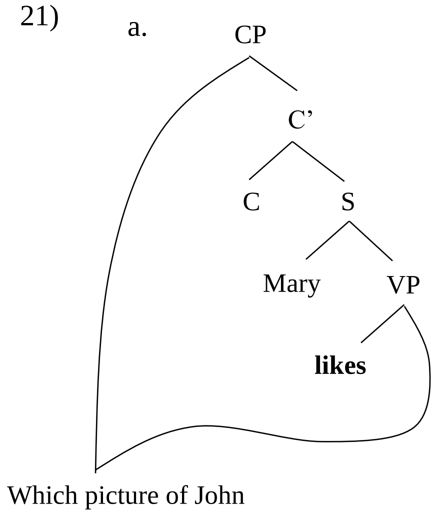
- An indexing rule that keeps track of occurrences of syntactic terms. It needs to be able to assign the same index to *John* and *himself*, but also to *which picture of himself* and *t*.

We will not deal in detail with these operations, in particular, a proper treatment of *wh*-dependencies are outside the scope of the present paper. For purposes of this work, we can summarise what we want the system to capture:

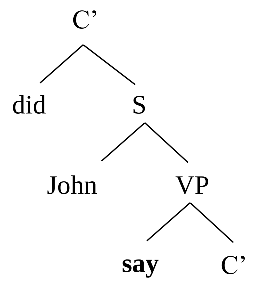
- The syntactic object *Which picture of himself* occurs in two syntactic contexts
- *John* and *himself* have the same semantic value

In a classical 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 (based on Frank, 2002: Chapter 5; Frank does not use multidominant trees, however. See Gärtner, 2002 for a formalisation of multidominance that is compatible with the approach pursued here), each of which contains exactly one lexical predicate (bolded):

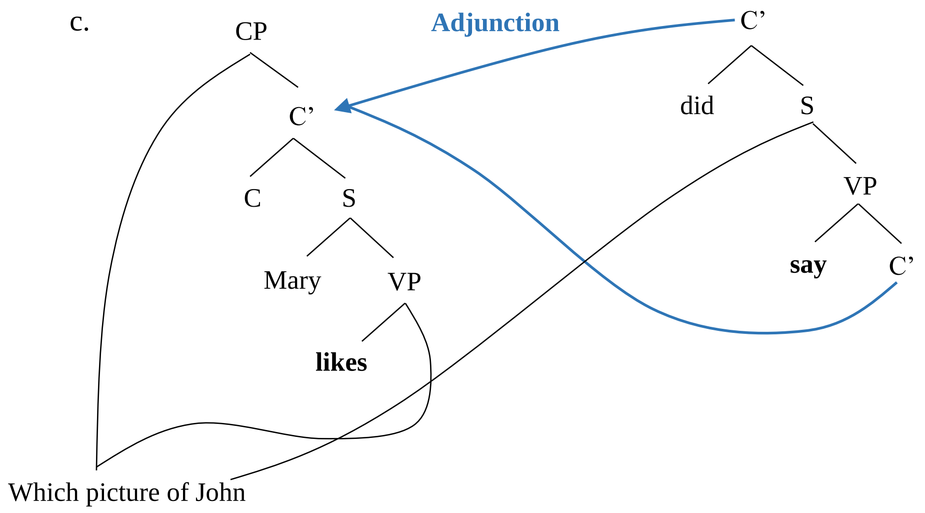
21) a.



b.



c.



A condition of TAGs is that links between displaced constituents and their 'base' position are preserved under adjunction (Joshi, 1985; Frank, 2002); in our terms, it means that the *distance* between a *wh*-operator and the variable it binds *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. In contrast, by virtue of multiplying objects in terms of copies or traces, Minimalism would assign a positive non-zero distance between the multiplied 'copies' of the operator complex (however that distance is measured), with relations being taken care of by some interface mechanism. Furthermore, the theory requires an additional operation to *delete* copies (e.g., in Chomsky, 2020a, b, anything that Minimal Search cannot find is marked as 'deletable') for reasons of 'computational efficiency' that remain unspecified; in any case, conditions on structure building and mapping are not linked to properties of the workspace.

Our approach, which capitalises on the identity property that is part of the definition of the workspace does not entail that the distance between *other nodes* and the operator complex *which picture of John* is 0, however. Let R be the root of the graph corresponding to the operator complex: then, defining a walk through the derived tree from any node P (P $\neq$ R) to R will give us a non-zero distance. Moreover, it is possible to define a unique sequence of nodes that get visited in that path, since the (elementary or derived) graphs are totally ordered (we will come back to this issue shortly).

A further condition imposed by grammar (in particular, by semantic interpretation) is that 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 as a function of 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; we follow Sarkar & Joshi's work in this respect). 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 suggests that *John* is an argument of *picture*). In a TAG, the elementary tree (21b) can be 'shoehorned' into (21a) at the node C' (which is at both the root and the frontier of (21b)); this yields a derived tree (21c) 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 result is analogous to what would be the output of *structure sharing* in Karttunen & Kay (1985), but the effects derive directly from the properties of the workspace as defined in **Section 2**. 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 dominance relations with nodes in each of the elementary trees[14].

[14] 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* between distinct elementary trees (i.e., between the neighbourhoods of distinct lexical predicates). 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 (similarly to Martin & Uriagereka, 2014). Note that the allowed dependencies are intimately connected to the definition of the workspace. In this case, for the passive *John was murdered*, we would be mapping a space illustrated in (a) into a space illustrated in (b):

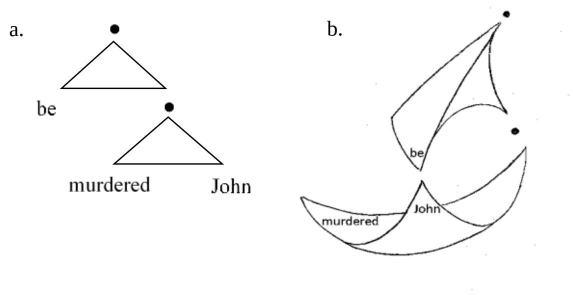
Let us now focus on the second point. We can express it as follows:

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

But we know that (22) can only hold if *John = himself* by (11b), which is equivalent to saying that in (22) *John* and *himself* are addresses to the same location in the workspace[15]. But this is, obviously, not necessary. We can have an example like:

23) 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 terminal string of the derivation and must define semantic interpretation as a function of that. This is the result of architectural assumptions: (i) the lexicon and the computational component are different spaces, and (ii) syntax is the only



In this view, pursued in Krivochen (2018) 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 whether 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). Note that if both instances of the word 'John' corresponded to the same object (as is required for semantic interpretation), we would be committed to saying that *John* both belongs and does not belong to the set {murdered, John}.

[15] This is 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 Gärtner (2014); Hornstein & Idsardi (2014), *inter alios*. 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.

generative component of the grammar, semantics being purely interpretive. Let us look at some simple cases:

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

The two *John*'s in (24a) do not refer to the same entity: they are two distinct addresses (corresponding, e.g., to John Lennon and John Paul Jones). Thus, they would be *repetitions*. In (24b) there would be a *copy* of *himself* in the position of Compl-V and another in the left periphery. In (24c) 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 local environment (Lees & Klima, 1963; McCawley, 1970; Grohmann, 2003: Chapter 3; Gärtner, 2014; Hornstein & Idsardi, 2014, among many others)[16], then a copy is needed (cf. Grohmann's *Copy Spell-Out*). At this point, if we turn to Chomsky (2020a), the scenario is quite unclear. Following Chomsky (2020a: 45) '[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 (Chomsky, 2020: 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, 2020a: 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 (see Müller, 2017 for a formulation of a structure removal operation that would be the inverse of Merge). Deletion operations presumably remove material from the workspace, but without a proper definition of *workspace* (which Chomsky does not provide) the whole exercise seems pointless. The third problem is that, if

[16] 'Local' means 'within a single elementary tree'. Specifically, for this particular case, an elementary tree that is the neighbourhood of a transitive verb and which does not include a passive auxiliary.

[…] *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)

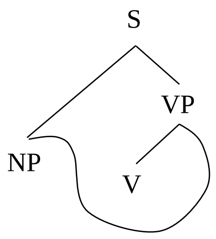
To account for (24b), in addition to MERGE creating two copies and leaving them both active we need an indexing mechanism: indexing tells us that *John* and *himself* denote the same entity while being two distinct elements in the set {John, {T, {admires, himself}}}. MERGE must create a copy of *himself* and introduce it back into the workspace at the root, without deleting or removing anything (deletion of the lower copy of *himself* would take place at PF). This would yield a representation like (25) (somewhat simplifying matters):

25) {himself, {C, {John, {T, {admires, himself}}}}}.

Again we are faced with the fact that *himself* both belongs and does not belong to the set {admires, himself}: it needs to belong to that set in order to satisfy the valency of *admire*, but not belong there so that it can be interpreted as a topic in the left periphery of the clause (see Gärtner, 2020 for detailed formal discussion about the inconsistencies between Chomsky's approach to copies and set-theoretic principles).

In the present view, the definition of workspace must play a role in accounting for these facts. What becomes relevant in (24a) 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 (24a) is not (cannot be) interpreted as (24c): 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 differentiate between (24a) and (24c); however, if we consider the definition of *neighbourhood* that we have been using we can say something interesting. Recall that in our approach syntactic operations relate points in the workspace, creating graphs in a metric space. The crucial fact about (25c) is that the same entity is the subject and object of *admire*. This can be captured with multidominance in a diagram along the general lines of (26), which follows strict immediate-constituency assumptions[17]:

26)



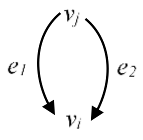
If grammatical functions are defined over structural configurations, in (26a) the node NP is directly dominated by the root and is the sister of VP, which makes it the Subject of the sentence, but it is also dominated by VP and is the sister of V, which makes it the Object (Chomsky, 1965: 69). Note that the requirement of a total order is respected: *dominance* is total, antisymmetric, and transitive. Explicitly, we can define a search sequence Σ in (27) under a breadth-first searching algorithm, which would be:

27) Σ = <S, NP, VP, V, NP>

In a configuration like (26), 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 (Condition A of binding theory); if the node address is called again from a *different* elementary tree, the Spell-Out would be pronominal (Condition B). 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

[17] The representation in (26) can in principle be further simplified if the nodes S and VP are eliminated, yielding a graph with parallel arcs as in (i):

(i)



Edges e_1 and e_2 are parallel iff $e_1<v_i, v_j> \wedge e_2<v_i, v_j>$. In this configuration, v_i dominates v_j establishing two different relations with it. In this case, if v_i is a nominal and v_j a verbal predicate, v_i is the verb's 1 and 2; thus, the verb is reflexive.

This representation is closer to Arc-Pair grammar than the one in (27), and requires grammatical functions to be primitives of the theory (Johnson & Postal, 1980; Postal, 2010). We will not provide arguments for or against the graphs in (26) and (i) (or other alternatives) here, leaving them as options, the choice between which should be empirically motivated.

(Reuland & Reinhart, 1993) can be captured without the need to invoke indices: what we need is a condition like (22), which is given by the definition of the workspace as a metric space. In this context, the difference between (24a), which in MGG terms would involve *repetitions* (of a word) and (24c), which would involve *copies* (assuming a transformational origin for true reflexive pronouns) is not something that an '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[18].

As a consequence of our definition of the syntactic workspace as a topological space, 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 (17a) (*which picture of himself did John say Mary likes?*). In a lexicalised grammar where every lexical head nucleates an elementary tree (and not just lexical predicates), as in Frank (2002, 2013), the noun *picture* would also head its own elementary tree. This 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 schematise the procedure as follows:

28) 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*

[18] 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 Gärtner, 2002, 2014; Johnson, 2016) to the 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 also made in Krivochen, 2015). The system proposed here does not require diacritics: either two 'John's are addresses to the same location in the workspace (and thus are the same object) or they are not. Our system is based on the idea that the workspace contains graphs rather than set, but even in a strict set-theoretic Minimalist approach the core issue seems to be defining how syntactic terminals are defined and what nodes in a phrase structure tree stand for (see also Gärtner, 2020).

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[19]

These need not be interpreted as sequential steps; because there is no separation between lexicon and syntax as two distinct spaces, the operations glossed in (28) bring some points in the space closer together, defining the neighbourhood of lexical heads. Thus, steps (28.1), (28.2), and (28.4) may proceed in parallel as they affect distinct regions of the workspace and none depends on the output of the others.

We said that elementary trees are sets of addresses and relations. For (17a) we define *two* elementary trees, corresponding to the neighbourhoods of 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 of dominance relations between nodes for the declarative version *John says*

[19] Relevantly, if syntactic objects in the workspace are assigned unique addresses (in other words: if addresses are unique identifiers for basic and derived expressions of the language), then some 'violations' of the principle of *Determinacy* discussed in the theoretical literature (Chomsky, 2019, 2020a; 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) $\{a, c\}, \{b, c\}$

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. 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 Relational Grammar / Arc Pair Grammar parlance (see Postal, 2010: 72 for a summary of grammatical relations and their symbolic designations), that c is the 2 of a and the 1 of b (as in raising-to-object constructions), for instance. A similar argument is, as far as we can see, valid for the other cases illustrated in Komachi et al. (2019: 280-281).

Mary likes a picture of himself (with Dependency Grammars, we assume that predicates always dominate their arguments; see Osborne et al. 2011):

29) 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 (thus the use of ⟨ ⟩): this is because the neighbourhoods of lexical predicates are strictly ordered graphs, not unordered sets. If nodes in the graph are categorematic basic expressions of the language (the expressions that are assigned addresses, also in Gorn, 1967), 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 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* dominated (transitively) by *like*. This is possible precisely because the grammar characterises strictly ordered directed graphs in the workspace, rather than building (unordered) sets.

This process of identification of identical addresses in distinct contexts follows from the characterisation of the workspace, and proves particularly useful to address the problem of distinguishing between *copies* and *repetitions*. The terms 'occurrence', 'copies', 'repetitions' have been used in a transformational framework (Collins & Groat, 2018; Chomsky, 1995, 2019, 2020a, b), but they are 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, 2020a, b) arise because derivations operate over sets of tokens lexical items and sets of sets of tokens of lexical items. For example, 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, 'context' here refers to the mother-daughter nodes of α. At the core of the problem there is a confusion, we think, between phrase markers as sets, phrase markers as graphs, and diagrams of phrase markers (see Postal, 2010: 7, ff.; McCawley, 1998: 47-48 for useful discussion about this point) that makes defining these objects and relations in an unambiguous way quite 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 (a full exploration of which is beyond the scope of this paper) 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 (26), for example, the node corresponding to the address for *John* is visited twice in a walk through the directed graph, 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 but avoided in graph-theoretic syntax. When neighbourhoods intersect via *substitution* or *adjunction*, identical addresses in initially distinct neighbourhoods are treated as a single element (similarly to Unification-based grammars and in contrast to Merge-based grammars), 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. In particular, we noted that different authors use the term *workspace* to mean different things, not always consistent with other parts of the formalism. Because the properties of the workspace are left mostly undefined in the current Minimalist literature, it is hard to link properties of the objects or operations to properties of the workspace. In this context, we needed to determine what exactly we want a *workspace* to do for us:

- Appealing to *workspaces* 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.
- 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. This results in a workspace-based theory of local relations.

In order to address these issues, the introduction of some mathematical concepts was unavoidable, and we believe also welcome. 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 features (desirably) topological distinguishability as well as variable distances between points (Willard, 2004; Munkres, 2000).

The second empirical component of this paper pertains to the analysis of simple cases of copies vs. occurrences. Conceiving of syntactic terms in structural descriptions as addresses to points in a space allows us to dispense with the multiplication of nodes in *chains*, for dependencies involving *copies*. Because local syntactic units are defined as graphs, whose nodes are points in the workspace, and these nodes are (in our particular implementation of this system) uniquely identifying addresses, it is possible to define structural descriptions that avoid set-membership paradoxes and simplify the system by dispensing with independent indexing mechanisms. Furthermore, we eliminate the distinction between a lexical space and a syntactic space: there is only *one* workspace which contains a set of strictly ordered directed graphs. 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*; see Stroik & Putnam, 2013) thus do not arise. The present view departs from MGG in adopting a graph-theoretic view instead of a set-theoretic view; properties of the former can be directly linked to properties of the workspace and (much research pending) seems to offer promising empirical advantages in the analysis of locality, long-distance dependencies, and co-reference.

Works cited

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

Boeckx, C. 2008. *Aspects of the syntax of agreement.* London: Routledge.

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

Bošković, Ž. 2020. On the Coordinate Structure Constraint, Across-The-Board movement, phases, and labelling. In *Recent Developments in Phase Theory*, ed. by J. van Craenenbroeck, C. Pots and T. Temmerman (eds.). Berlin: De Gruyter.

Branan, K. & M. Yoshitaka Erlewine (2021) Locality and (minimal) search. To appear in *Cambridge Handbook of Minimalism*, ed. by K. K. Grohmann & E. Leivada (eds.). https://ling.auf.net/lingbuzz/005791

Bravo, A., L. García Fernández & D. G. 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

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

Chomsky, N. 1977. Conditions on transformations. In *Essays on Form and Interpretation*. New York: North Holland.

Chomsky, N. 1994. Bare phrase structure. *MIT occasional papers in linguistics* 5.

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

Chomsky, N. 2000. Minimalist Inquiries: the framework. Minimalist Inquiries: The Framework. In *Step by Step – Essays in Minimalist Syntax in Honor of Howard Lasnik*, ed. by R. Martin, D. Michaels and J. Uriagereka. Cambridge, Mass.: MIT Press. 89-155.

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

Chomsky, N. 2008. On Phases. In *Foundational issues in linguistic theory*, ed. by R. Freidin, C. Otero, and M. L. Zubizarreta. Cambridge, Mass.: MIT Press. 133-166.

Chomsky, N. 2012. Foreword. In *Phases: Developing the Framework*, ed. by A. Gallego. Berlin: Mouton de Gruyter. 1-7.

Chomsky, N. 2013. Problems of Projection. *Lingua* 130. 33-49.

Chomsky, N. 2015. Problems of Projection: Extensions. In *Structures, Strategies and Beyond: Studies in honour of Adriana Belletti*, ed. by E. Di Domenico, C. Hamann & S. Matteini. Amsterdam: John Benjamins. 1-16.

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

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

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

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

Cinque, G. 2004. 'Restructuring' and Functional Structure. In *Restructuring and Functional Heads. The Cartography of Syntactic Structures*, ed. By G. Cinque.. Oxford: OUP. 132–192.

Citko, B. 2011. Multidominance. In *Handbook of Linguistic Minimalism*, ed. by C. Boeckx. Oxford: OUP. 119-142.

Collins, C. 2002. Eliminating Labels. In *Derivation and Explanation in the Minimalist Program*, ed. by S. D. Epstein and T. D. Seely. Oxford: Blackwell. 42-64.

Collins, C. 2017. Merge(X, Y) = {X, Y}. In *Labels and Roots*, ed. by L. Bauke & A. Blühmel. Berlin: Walter de Gruyter. 47-68.

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

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

Dowty, D., R. Wall, & S. Peters. 1981. *Introduction to Montague Semantics*. Dordrecht: Kluwer.

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

Epstein, S.; H. Kitahara & T. D. 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, S.; H. Kitahara & T. D. Seely. 2020. Unifying Labeling under Minimal Search in "Single-" and "Multiple-Specifier" Configurations. *Coyote Papers: Working Papers in Linguistics*, 22. 1-11.

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

Frank, R. 2013. Tree adjoining grammar. In *The Cambridge Handbook of Generative Syntax*, ed. by M. Den Dikken. Cambridge: CUP. 226-261.

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

Gavryushkin, A. & A. J. Drummond. 2016. The space of ultrametric phylogenetic trees. *Journal of Theoretical Biology* 403. 197-208.

Gärtner, H-M. 2002. *Generalized transformations and beyond. Reflections on Minimalist syntax*. Berlin: Akademie Verlag.

Gärtner, H-M. 2014. Strange Loops: Phrase-Linking Grammar Meets Kaynean Pronominalization. *Linguistische Berichte* 24. 1-13.

Gärtner, H-M. 2020. Copies from 'Standard Set Theory' - A Remark on Chomsky, Gallego & Ott (2019). https://ling.auf.net/lingbuzz/005942

Gorn, S. 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, K. 2003. *Prolific domains: on the anti-locality of movement dependencies*. Amsterdam: John Bejnamins.

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

Heim, I. & A. Kratzer. 1998. *Semantics in Generative Grammar*. Oxford: Blackwell.

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

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

Hornstein, N. & W. Idsardi. 2014. A Program for the Minimalist Program. In *Minimalism and Beyond: Radicalizing the interfaces*, ed. by P. Kosta, S. L. Franks, T. Radeva-Bork & L. Schürcks. Amsterdam: John Benjamins. 9-36.

Hughes, B. 2003. Trees and ultrametric spaces: a categorical equivalence. *Advances in Mathematics* 189. 148–191.

Jackendoff, R. 2011. Alternative minimalist visions of language. In *Non-transformational Syntax*, ed. by R. D. Borsley and K. Börjars. Oxford: Wiley-Blackwell. 268-296.

Jayseelan, K. 2017. Parallel Work Spaces in Syntax and the Inexistence of Internal Merge. In *Perspectives on the architecture and the acquisition of syntax*, ed. by G. Sengupta, S. Sicar, M. Gayathri Raman & R. Balusu. Singapore: Springer. 115-136.

Johnson, D. & P. Postal. 1980. *Arc Pair Grammar*. Princeton, NJ.: Princeton University Press.

Johnson, K. 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, A. K. 1985. Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? In *Natural Language Parsing*, ed. by D. Dowty, L. Karttunen, and A. Zwicky. Cambridge, Mass.: CUP. 206-250.

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

Joshi, A. & Y. 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, I. 1977. *Set theory and metric spaces*. Boston: Allyn and Bacon.

Karttunen, L. & M. Kay. 1985. Structure sharing with binary trees. *Proceedings of the 23rd Association for Computational Linguistics*. Chicago, Illinois: University of Chicago. 133-136.

Kato, T.; H. Narita, H. Kasai, M. Zushi and N. Fukui. 2016. On the primitive operations of syntax. In *Advances in Biolinguistics*, ed. by K. Fujita & C. Boeckx. London: Routledge. 29-45.

Kayne, R. 1984. *Connectedness and Binary Branching*. Dordrecht: Foris.

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

Kitahara, H. 2020. ‘Multiple specifier’ configurations revisited. *Reports of the Keio Institute for Cultural and Linguistic Studies* 51. 207-216.

Komachi, M., H. Kitahara, A. Uchibori, and K. Takita. 2019. Generative procedure revisited. *Reports of the Keio Institute of Cultural and Linguistic Studies* 50 (2019), 269-283.

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

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

Krivochen, D. G. 2018. *Aspects of emergent cyclicity in language and computation*. PhD dissertation, University of Reading.

Krivochen, D. G. 2021. The search for Minimal Search. Under review. https://ling.auf.net/lingbuzz/005817

Krivochen, D. G. & L. 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

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

Lakoff, G. 1976. Pronouns and Reference. In *Notes from the linguistic underground*, ed. by J. D. McCawley. New York: Academic Press. 275-336.

Lasnik, H. 2009. Shortest Move and Equidistance. Linguistics 611 class notes, University of Maryland.

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

Manzini, M. R. & L. Savoia. 2011. *Grammatical categories: Variation in Romance languages.* Cambridge: CUP.

Martin, R. & J. Uriagereka. 2014. Chains in Minimalism. In *Minimalism and Beyond: Radicalizing the interfaces*, ed. by P. Kosta, S. L. Franks, T. Radeva-Bork and L. Schürcks. Amsterdam: John Benjamins. 169-194.

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

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

McCawley, J. 1970. Where do Noun Phrases come from? In *Readings in English Transformational Grammar*, ed. by R. Jacobs & P. Rosenbaum. Waltham: Ginn & Co. 166-183.

McCawley, J. 1982. Parentheticals and Discontinuous Constituent Structure. *Linguistic Inquiry*, 13(1). 91-106.

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

Medeiros, D. P. & M. Piattelli-Palmarini. 2018. The Golden Phrase: Steps to the Physics of Language. In *Language, Syntax, and the Natural Sciences*, ed. by A. Gallego & R. Martin. Cambridge: CUP. 333–350.

Murtagh, F. 2004. On Ultrametricity, Data Coding, and Computation. *Journal of Classification* 21. 167-184.

Morin, Y-C. & M. O'Malley. 1969. Multi-rooted vines in semantic representation. In *Papers from the Fifth Regional Meeting of the Chicago Linguistic Society*. University of Chicago. 178-185.

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

Müller, G. 2004. Phrase Impenetrability and Wh-Intervention. In *Minimality Effects in Syntax*, ed. by A. Stepanov, G. Fanselow, & R. Vogel. Berlin: Mouton/de Gruyter. 289-325.

Müller, G. 2017. Structure Removal: An Argument for Feature-Driven Merge. *Glossa* 2(1). http://doi.org/10.5334/gjgl.193

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

Ojeda, A. 1987. Discontinuity, Multidominance, and Unbounded Dependency in GPSG. In *Syntax and Semantics 20: Discontinuous Constituency*, ed. by G. Huck & A. Ojeda. New York: Academic Press. 257-282.

Osborne, T., M. T. Putnam & T. Groβ. 2011. Bare phrase structure, label-less trees, and specifier-less syntax: Is Minimalism becoming a dependency grammar? *The Linguistic Review* 28. 315-364.

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

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

Rammal, R, G. Toulouse, and M. A. Virasoro. 1986. Ultrametricity for Physicists. *Reviews of Modern Physics* 58(3). 765-788.

Reid, M. & B. Szendröi. 2005. *Geometry and Topology*. Cambridge: CUP.

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

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

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

Saddy, D. 2018. Syntax and Uncertainty. In *Language, Syntax, and the Natural Sciences*, ed. by A. Gallego & R. Martin. Cambridge: CUP. 316-332.

Sarkar, A. & A. 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

Schmerling, S. F. 1982. How imperatives are special, and how they aren't. In *Papers from the parasession on nondeclaratives*, ed. by R. Schneider, K. Tuite, and R. Chametzky. Chicago: Chicago Linguistic Society. 202–218.

Seely, T. D. 2006. Merge, Derivational C-command, and subcategorization in a label-free syntax. In *Minimalist Essays*, ed. by C. Boeckx. Amsterdam: John Benjamins. 182-217.

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

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

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

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

Turing, A. 1936. On Computable Numbers, with an Application to the Entscheidungsproblem, *Proc. London Math. Soc*. 42(2). 230-265.

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

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

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

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

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

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

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

Zwicky, A. & S. 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