Remarks on Primary Merge: A reply to De Belder and Van Craenenbroeck (2015)

Ivona Kučerová and Adam Szczegielniak

De Belder and Van Craenenbroeck (2015) (B&C) argue, based on a variety of data and cited literature, that roots have: (i) no syntactic category, or grammatical features, (ii) that they are defined structurally rather than by root features, and (iii) that they are always dominated by functional material. In order to derive these properties of roots, B&C propose a novel approach to the derivational grammar that makes three crucial assumptions: (i) Primary merge involves a null set being concatenated with a functional head; the null set is a placeholder for Late Insertion of the root. (ii) Merge is unary. (iii) Unary merge forms ordered sets which are the equivalent of Chomsky (2013)'s labelled sets.

B&C's paper is an important and much needed undertaking aimed at formalizing some of the most basic assumptions concerning the architecture of the grammar. Unfortunately, the system over-generates root positions. As we show, the system enforces a zero set as a sister of T which makes empirically incorrect predictions for the structure of the clausal (CP-TP-vP-VP) spine. Moreover, we show that B&C's proposal to replace binary merge with unary merge creates an inherent contradiction within the application of primary merge. We identify bottom up unary merger as the source of these contradictions and argue that rejection of Chomsky's binary merge in favor of unary merge leads to irreparable problems when identifying conditions for structure building operations. We thus conclude that B&C's system cannot replace Chomsky's binary merge architecture, nor can it account for the distribution of roots.[1]

1 De Belder and Van Craenenbroeck (2015)'s proposal

The primary structure building operation in B&C's system is *unary merge*. The adoption of asymmetric unary merge rejects Chomsky (1995)'s proposal that merge is *binary*, i.e., a symmetrical operation that takes two arguments (units or sets), and

[1]In this reply we only examine the core properties of the system; we do not explore B&C's analysis of compounds and of other phenomena, nor do we reject the basic empirical observations that motivate B&C's proposal.

produces a new set with two symmetrical, i.e., unordered, elements. Unary merge has its origins in the work of Jaspers (1998), Fortuny (2008) and Zwart (2009a,b, 2011). In lieu of the binary merge operation, B&C propose to adopt a new definition of merge, defined as:

(1) Merge selects a single subset from a resource (e.g., $\{\alpha\}$), includes it in the object under construction (δ), and yields an ordered pair (e.g., $<\{\alpha\}, \delta>$), assuming that $\{\alpha\}$ projects). (De Belder and Van Craenenbroeck, 2015, 636)

In order to make the definition in (1) complete, one has to define the terms *resource* and *object under construction*. Loosely speaking, for B&C a resource is a subset of the lexicon that becomes the domain (input) of the function merge. The resource is sometimes also called the array, or numeration. B&C understand the object under construction, δ, to be broadly defined as the output set (domain) that has been created by the immediately preceding iteration of merge. In a nutshell, unary merge is a single argument function whose domain is a subset of the lexicon (resource/array/numeration) and whose iteratively defined image (output) are the contents of the workspace (object under construction), as in Figure 1.[2]

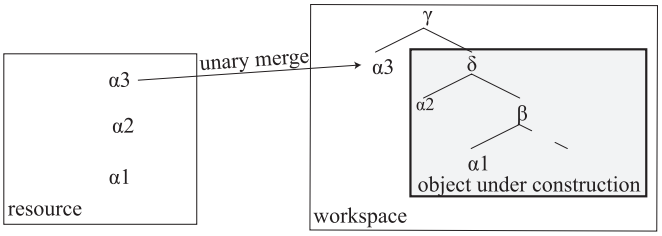

Figure 1: Unary merge taking α_3 from the resource and merging it with the object under construction δ to form an ordered set $<\alpha_3,\delta>$ whose label $\gamma=\alpha_3$

1.1 Primary merge

Having outlined the core architecture of B&C's proposal we can now describe the underlying proposal that every lexical/spell out domain has a null set at its core, and that this null set serves as a placeholder for the root. B&C propose that primary unary asymmetrical merge involves adding an element, α, from the resource (subset of the lexicon) to the unary merge object under construction, δ (the workspace), in order to form an ordered set $<\alpha, \delta>$. Following Zwart (2009a, 2011), B&C claim that, in the case of primary unary merger, the object under construction is the zero set $(\delta) = \emptyset$ since there were no previous iterations of unary merge. Let us highlight this claim in B&C's own words, "when an element $\{\alpha\}$ is the first one to be taken from the

[2]B&C propose that labeling of new sets can be achieved by having unary merge generate intrinsically ordered sets.

resource by unary merge, it is included into an empty derivational workspace; that is, the object under construction is the empty set ∅." (De Belder and Van Craenenbroeck, 2015, 636). The implication being the object under construction is a null set, and so is the workspace. Thus, the basis of every lexical domain is an ordered pair where one element is the zero set, and the other is a lexical item from the resource, which projects, giving us an ordered set $\{\alpha\ \{\alpha, \emptyset\}\}$. Subsequent applications of unary merge add elements from the resource, but the zero set remains embedded at the bottom of the root. See Figure 2.

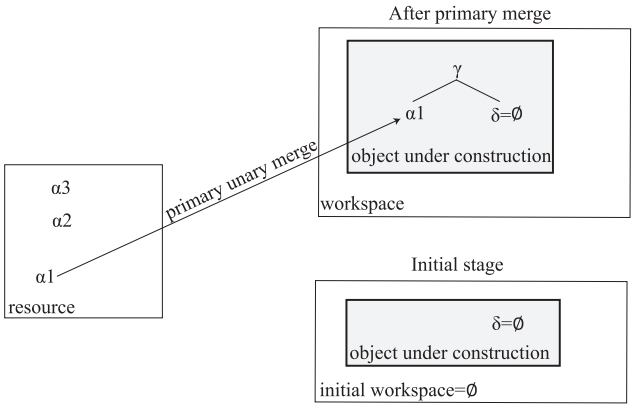

Figure 2: Primary unary merge taking α_1 from the resource and merging it with the object under construction δ=∅ to form an ordered set $<\alpha_1,\delta>$ whose label $\gamma=\alpha_1$

Crucially for B&C, the zero set "is a syntactic terminal that receives phonological exponence in the postsyntactic morphological module." (De Belder and Van Craenenbroeck, 2015, 639). To be precise, the zero set is a *placeholder* for Late Insertion of roots and inherently derives the fact that roots have no grammatical features or category, are defined structurally not lexically, roots are always merged below functional heads.

2 Clausal spine derivation

B&C argue that each lexical domain/derivational workspace starts with a zero set merged with a functional projection. For B&C the vP is a phase and a separate derivational workspace. Each DP argument contained within a vP is also a separate derivational workspace, and a phase. Completion of a derivational workspace whose contents are to be embedded in a larger derivational workspace, such as a DP complement embedded in a vP, requires the *re-admittance* of a DP complement to the resource prior to the construction of the matrix vP workspace. B&C assume that re-admittance of a DP complement to the resource creates a new derivational workspace that is null, and that this null workspace contains a null object under construction.

This allows the matrix vP domain to be constructed with an embedded null set that corresponds to the initial null object under construction, which subsequently becomes the V root position.

It means that once vP derivational domain is complete, a new derivational domain with a new derivational workspace needs to be constructed. By definition, every new derivational workspace is empty and, as such, it contains a null object under construction. The empty set that is the null object under construction becomes the root placeholder after primary merge.[3] Let us consider B&C's simplified derivation of transitive vP, such as '*I ate the cookie*', which, for expository purposes, omits any functional heads other than D and v. In their derivation of a vP derivational workspace, B&C assume that the external argument is derived first because subjects are islands and re-admittance to the resource creates *opacity*. The DP merged with a null set, which is the empty workspace containing an empty object under construction and the external DP is spelled out and 'readmitted' to the resource, creating a new derivational workspace, as in Figure 3.

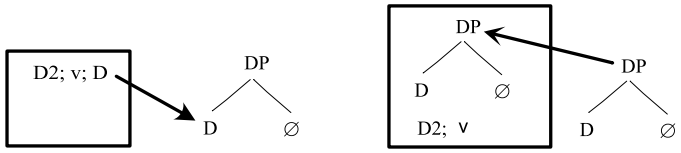

Figure 3

B&C, in order to account for opacity effects of the v-V base, derive the v+root structure prior to the derivation of the internal argument, and have the v+null root position readmitted to the resource, as in Figure 4.

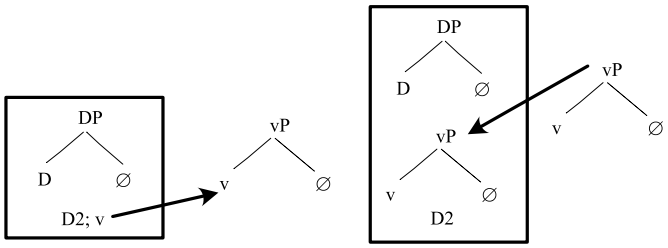

Figure 4

The internal argument is then built. Note that since vP has been readmitted to the resource, the workspace is empty and D2 is merged again with a null object under construction inside a null workspace to form D2P – the inner argument, as in Figure 5.

[3]One Derivational Workspace, One Root (ODWOR): "In every derivational workspace, there is exactly one root, and for every root there is exactly one derivational workspace." (De Belder and Van Craenenbroeck, 2015, 642)

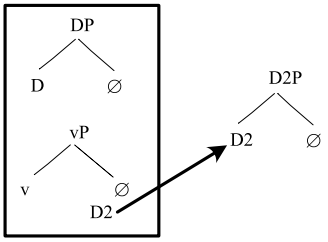

Figure 5

Instead of being readmitted, again, to the resource, the inner argument D2P is merged with the vP from the resource, as in Figure 6.

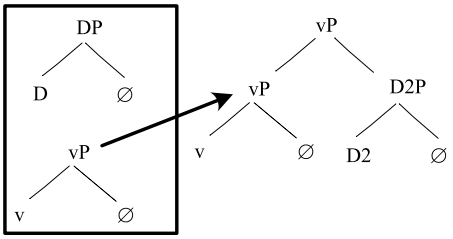

Figure 6

Finally, the outer argument is merged with the vP, as in Figure 7.

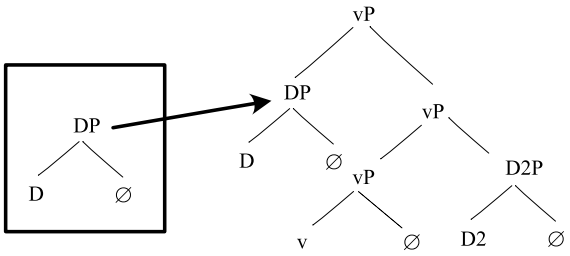

Figure 7

Even though the relationship between a derivational workspace and spell-out is not detailed in B&C's system, in the derivations they provide, spell out, which is triggered by the insertion of the highest head of a given domain (Bošković, 2014), is accompanied by the creation of a new derivational workspace. B&C use the term 'layering' when a DP is constructed first and then readmitted to the resource in order to allow for the construction of the vP in a new derivational workspace, in their own words: "note that {v} cannot be (Primary-)Merged to the empty set either, as the derivation is no longer in its null state (…). The solution lies in layering the derivation – that is, in spelling out the structure (…) and readmitting the object thus constructed to the resource (…). This re-admittance operation has cleared the

workspace, and hence makes possible a new application of primary merge" (De Belder and Van Craenenbroeck, 2015, 644).[4]

Crucially for us, B&C argue that there is an intimate relationship between: (i) ending a lexical domain, understood as merging the last functional head associated with a given derivational domain, (ii) spell out, which is triggered by ending a lexical domain, and (iii) re-admittance to the resource, which always accompanies spell out. When construction of a structure, such as a vP, exhausts all the functional projections associated with vP, spell out needs to proceed, which means a new derivational workspace is needed.

The vP needs to be spelled out in order to be a phase. If the vP is spelled out then it is readmitted to the resource.[5] Readmission to the resource forces any subsequent merger to be primary merge with a zero object under construction that has its source from a zero workspace. However, the need for primary merge with a zero object under construction poses a problem for any subsequent merge of a functional category that selects a vP as its sister. In short, the proposal *over-generates* by forcing T (or any other relevant functional category) to have its own empty root position.

Let us extrapolate the derivations in B&C to highlight the nature of the problem. Let us assume that the resource contains T and C heads and that a ditransitive vP has been constructed and is being sent to spell out which requires that it is also readmitted to the resource, as in Figure 8.

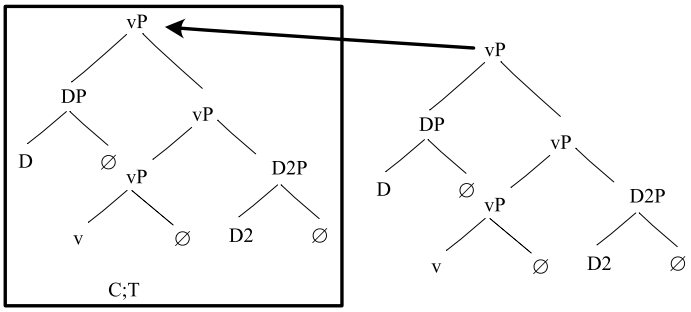

Figure 8

The workspace is null and contains a null object under construction. The only way to introduce T into the structure is to merge T with a null object under construction, as in Figure 9.

[4]However, the creation of a new workspace does not necessarily trigger spell out. As we saw, in order to account for opacity effects, their account of simple transitives allows the v+root constituent to be readmitted to the resource prior to the insertion of the highest head of the vP domain.

[5]Not readmitting the vP to the resource, but spelling it out would break the relationship between readmission to the resource and spell-out. Considering that B&C argue that opacity to syntactic computations is generated via readmission and spell-out, it would have to be argued that vP is less of phase than a DP. However, B&C assume vP is a phase en par with DPs.

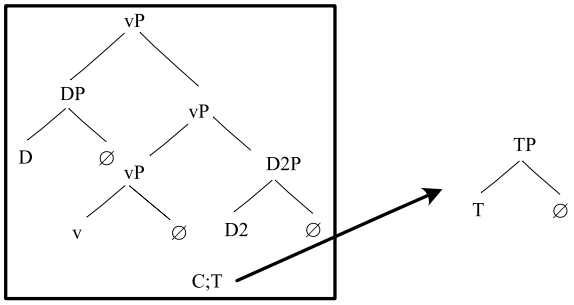

Figure 9

This is already problematic since TP domains are assumed to be rootless. Any subsequent merger of the vP will require it to be merged with TP complex which has a null root placeholder at its base, as in Figure 10.

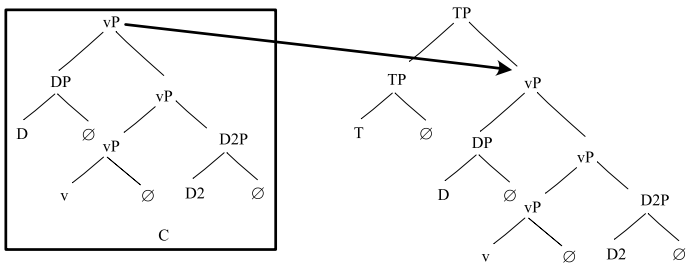

Figure 10

The prediction that T has a vacant root position does not find any empirical support. Any additional mechanisms that would allow the elimination of such null root position would weaken this proposal significantly since of its main strengths is that these null root placeholders are devoid of any features. This is claimed to be a major advantage over Distributed Morphology models that advocate root marking. As B&C (649) state "while in our model root terminal nodes are radically devoid of features, in DM these positions are explicitly marked as such." B&C argue that root positions devoid of any features have the advantage that they can accommodate functional heads as roots. However, if we were to allow to distinguish the zero set that is the sister of T from other zero sets (such as sister of D or of v), then we would indeed have to have features on zero sets. However, adding any diacritic or feature would take away the original conceptual advantage of the proposal that assumes null sets are devoid of any distinguishing properties.

The above issue does not have any clear obvious solution. However, in addition to the empirical problem of over-generation, the proposed system involves an irreparable contradiction, closely connected to the empirical problem we highlighted in this section. Next section discusses the contradictory property of the system in detail.

3 An inherent Contradiction: the issue with embedding a zero set inside a zero set

The first problem with the outlined model is that it fails to derivationally distinguish a zero set that is the workspace from a zero set that is the object under construction. Let us cite again the crucial claim in B&C : "when an element $\{\alpha\}$ is the first one to be taken from the resource by unary merge, it is included into an empty derivational workspace; that is, the object under construction is the empty set $\emptyset$." (De Belder and Van Craenenbroeck, 2015, 636). Let us dissect the proposed initial steps of any syntactic derivation. We have a workspace (w) that is the zero set. We also have an object under construction which is also the zero set. The assumption in B&C is that the object under construction is somehow contained or is part of the workspace: $\delta \subset$ w. In the initial stage, both δ and w are null.

However, there is a problem with such reasoning. If we do not distinguish the workspace from the object under construction then the implication is the workspace and object under construction are the same: δ=$\emptyset$ w=$\emptyset$ $\therefore$ δ=w. This is one of the basic tenets of set theory (Fraenkel et al., 1973; Zermelo, 2002): there is only one zero set within a given class/type, and, by definition, sets that are null are equal to each other.[6] What we need to is to distinguish a null workspace from a null object under construction. This can be achieved in two ways. Either we derivationally introduce a null object under construction by assuming there is an operation responsible for its presence, or we axiomatically assume the existence of a null object under construction by postulating that an object under construction is of a different type/class than a workspace.[7] The second option is uninteresting since it essentially axiomatically postulates the presence of a null root position independently of the type of structure building operations.[8] Unfortunately, the first option is impossible to implement in B&C's system since the only operation that can generate an object under construction is unary merge. However, primary unary merge, which is a single argument iterative operation, cannot be responsible for the presence of the initial zero set since it requires the presence of an object under construction prior to its application. The proposal in B&C glosses over this problem which unfortunately has dire consequences for the derivation. In their account unary primary merge selects an element α from the resource and merges it with δ. However, if we follow B&C and assume that initially

[6] The properties of the unique zero/empty set are as as follows:

$\forall$A: $\emptyset \subseteq$ A	Any set {A} has the empty set as its subset.
$\forall$A: A $\cup \emptyset$ = A	Union of the empty set with set {A} equals the set {A}
$\forall$A: A $\cap \emptyset = \emptyset$	Intersection of the empty set with the set {A} equals the {A}
$\forall$A: A $\times \emptyset = \emptyset$	Cartesian product of the empty set with the set {A} is the empty set
$\forall$A: A $\subseteq \emptyset \Rightarrow$ A = $\emptyset$	The only subset of the empty set is the empty set
$\mathcal{P}\{\emptyset\} = \{\emptyset\}$	Power set of the empty set is the empty set

[7] We assume that a null workspace automatically generates a null object under construction then objects under construction are not solely derived via merge.

[8] See Guimarães (2004, 221ff)'s starting axiom, which B&C briefly mention.

δ=w, then primary unary merger forms and object <α, δ> which by definition equals <α,w>. This is because <α, δ>= <α,w> by virtue of δ=w. And since for B&C, δ=$\emptyset$ even after primary merge, then we have <α, $\emptyset$>= <α,w> meaning w=$\emptyset$ even though it contains α. This is obviously a contradiction and an undesired result. That is why in order to claim that <α, δ>$\neq$<α,w>, we need to assume that δ and w are distinct objects, even when they are both null. As we already mentioned, there are numerous ways of doing that. Within a generative model, the most natural way is to generate the null object under construction inside the null workspace, thus making it derivationally on object of a different type than the workspace itself, which would allow us to keep it null even when the workspace is no longer null. However, this option is not possible in an architecture where there is unary merge.

3.1 Issue with recursion

We argue that the problem with defining initial conditions for primary merge stems from the fact that unary merger is *iterative* in B&C's system when it should be *recursive*. A single argument operation where n=n+1, for n ranging from $-\infty$ to $+\infty$, is simply an iterative infinite loop with no staring point and no end. There is no starting point since for every n there is an n$-$1 prior to it, and there is no end since for every n there is an n+1 after it. Unconstrained unary merge has the structure of such a infinite loop: it takes an argument α, adds it to δ to form δ, hence δ=δ+α. Without clearly defined initial constraints, unary merge becomes an infinite loop with no lower bound. An infinite loop is not a recursive structure, at least not the kind we are looking for until its initial conditions are defined in a way that allows us to set the lower bound of merge.

B&C's proposal has the operation unary merge serving as a successor function (von Neumann 2002, cf. Chomsky 1957) with no well-defined initial state. Strictly speaking, the presence of $\delta = \emptyset$ inside a workspace cannot be just stipulated as an axiom, since B&C's whole project is to justify, based on independently motivated assumptions about the nature of merge, the presence of the zero set at the bottom of every lexical domain. If we are to stipulate that a null δ is always contained in a null workspace w prior to primary merge, then we might as well just stipulate that the root position is always empty. Otherwise, one ad hoc stipulation is replaced by a different one. To sum thing up, a unary merge approach either forces us to live with the contradiction that the object under construction is the workspace, but does not equal it later on, leading to contradictory results, or we end up having infinite loops, instead of recursive merge. Neither result is desirable.[9]

[9]Zwart (2011)'s proposal differs from B&C in that it involves top down derivations. That the zero set is at the bottom of each tree is the end result of a derivation, not its initial state. In other words, a null set is the upper bound of that derivation; it is as if we were counting down from $+\infty$ to $\emptyset$. Such an approach avoids problems in deriving primary merge since merger is a decomposition function that terminates when there is nothing left to decompose.

4 Conclusion

De Belder and Van Craenenbroeck (2015) raise important questions and observations about the nature of roots, and the need to have a formalized model of the grammar as far as primary merge. The architecture of grammar that they propose aims to generate via syntactic derivation morpho-syntactic root properties. We are confident that this is the right approach. However, we show that their model based on unary merge suffers from an inherent flaw that does not allow to derivationally distinguish null sets. Thus a null set that isa workspace is not distinguished from the null set that is the object under construction contained in that null workspace. Moreover, it is impossible to distinguish different null workspaces within the CP/TP-vP/VP spine, where the CP/TP part of the spine is not endowed with a null root marker, but the vP/VP part is. The aim of this paper is not to provide solutions to these issues but to highlight problems in De Belder and Van Craenenbroeck's proposal with the hope that future research will nt only build on their proposal, but also modify it so that it is free of its current drawbacks.

References

Bošković, Željko. 2014. Now I'm a phase, now I'm not a phase: On the variability of phases with extraction and ellipsis. *Linguistic Inquiry* 45:27–89.

Chomsky, Noam. 1957. *Syntactic structures.*

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

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

De Belder, Marijke, and Jeroen Van Craenenbroeck. 2015. How to merge a root. *Linguistic Inquiry* 46:625–655.

von Neumann, John. 2002. On the introduction of transfinite numbers. In *From Frege to Gödel: A source book in mathematical logic, 1879–1931*, ed. Jean van Heijenoort, 346–354. Cambridge, Mass.: Harvard University Press.

Fortuny, Jordy. 2008. *The emergence of order in syntax.* Amsterdam: John Benjamins.

Fraenkel, Abraham Adolf, Yehoshua Bar-Hillel, and Azriel Levy. 1973. *Foundations of set theory*, volume 67 of *Studies in Logic.* Amsterdam: Elsevier.

Guimarães, Max. 2004. Derivation and representation of syntactic amalgam. Doctoral Dissertation, University of Maryland, College Park.

Jaspers, Dany. 1998. Categories and recursion. *Journal of Applied Linguistics* 12:81–112.

Zermelo, Ernst. 2002. Investigations in the foundations of set theory. In *From Frege to Gödel: A source book in mathematical logic, 1879–1931*, ed. Jean van Heijenoort, 199–215. Cambridge, Mass.: Harvard University Press.

Zwart, C. Jan-Wouter. 2009a. Prospects for top-down derivation. *Catalan Journal of Linguistics* 8:161–187.

Zwart, C. Jan-Wouter. 2009b. Uncharted territory? Towards a non-cartographic account of Germanic syntax. In *Advances in comparative germanic syntax*, ed. Jorge Hankamer Artemis Alexiadou, Thomas McFadden, Justin Nuger, and Florian Schäfer, 59–84. Amsterdam: John Benjamins.

Zwart, C. Jan-Wouter. 2011. Structure and order: Asymmetric merge. In *Oxford handbook of linguistic Minimalism*, ed. Cedric Boeckx, 96–118. Oxford – New York: Oxford University Press.