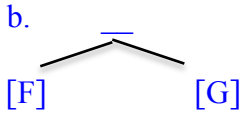
Labelling: reinstalling 'projection by selection'

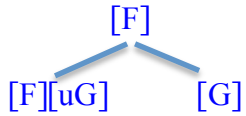
Question: Since Chomsky (2013) labelling has become a widely discussed topic within minimalist syntax. The central question has been why and how the merger of F and G, {F, G}, should receive a label. What determines what needs to be included in the __ slot in (1)?

(1) {__ , {[F], [G]} } or b. 

Background: As is well known, a strong tradition in syntax took projection to follow from selection. If a determiner selects an NP, it is D (and not N) that projects. Similarly, if a particular head selects a second element (to be its specifier), again this head, via an intermediate bar-level, projects up to the top node. In a proto-typical X-Bar projection, X projects, because X selects its complement and its specifier. Naturally, the question arises as to why the selecting element should also project its label. Why is it that if F selects for G, the label of the merger is F? And how do such systems apply to instances of adjunction, where, if any selection plays a role at all, it is the adjunct that selects its modifiee rather than the reverse? As these questions have not been satisfactorily addressed in the literature, alternative proposals to labelling have been proposed that do not reduce labelling to selection.

Proposal: Current popular labelling algorithms emerged to solve the paradox that (i) no inherent property of Merge requires the result of the merger to be associated with a particular feature and (ii) that every node in the tree (either in narrow syntax, or at the interfaces) needs to carry a label. In essence, Chomsky's (2013, 2014) proposal is to allude to independently available mechanisms (Minimal Link, feature sharing, etc.) to assign labels to unlabelled mergers. In this paper, I start with the argument that premise (i) is incorrect, and that, consequently, there is no paradox that needs to be resolved anymore. To see this, take (1) again. If (1) is the merger of two elements, one carrying [F] and the other carrying [G], the true question to be addressed is why the merger of (1) is not the union {[F], [G]}. If no particular principle in the syntax is responsible for removing particular features (or in other words: for the loss of syntactic information), the default hypothesis should be that all features percolate. If only one of the two features, say [F], percolates and becomes the label, the true question to address is not what special property [F] must have such that [F] projects; instead the true question should be why [G] does not percolate?

I argue that the latter question can be straightforwardly addressed if, following Zeijlstra (2014), so-called uninterpretable formal features are to be checked in the syntax, as they encode syntactic dependencies. Concretely, I take a feature [uG] on [F] to simply encode a dependency that [F] needs to merge with an element carrying a feature [G]. Formal features thus determine the distributional properties of syntactic elements. An advantage of this feature architecture is that selectional features reduce to uninterpretable features as well. If an element F selects for an element G(P), F just carries a feature [uG].

(2) 

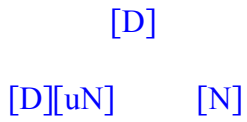
Application: Now let's see what happens if an element with a feature bundle {[F], [uG]} merges with an element with the feature bundle {[G].s shown in (2), the element carrying [F][uG] is an element carrying [F] that still needs to be merged with an element an element of category [G]. But that means that the categorial properties of the top node are different in the sense that the left sister is an element of category F that needs to undergo merger with am element of category G, whereas the top node is simply an element of a category F that lacks this need: in other words, it lacks a feature [uG]. But since the element carrying [G] fulfilled this particular categorial need, [G] can no longer remain active in doing so and should not be available at the top node either.

The features of the top node are nothing but the result of the left sister acting as a function that applies to the right-sister. In this sense, the proposal is very similar to proposals made in versions of categorial grammar (where G\F applied to G would result in F), or type theory (<e,t> applied to e yields t). Hence we can formulate the following rule:

(3) **Rule:** if a feature bundle α merges with a feature bundle β, every feature that is an element of α or β percolates (and becomes part of the feature bundle of the top node), unless some interpretable feature [F] part of α has a matching uninterpretable feature [uF] part of β, or the other way round. In that case neither [F], nor [uF] percolates.

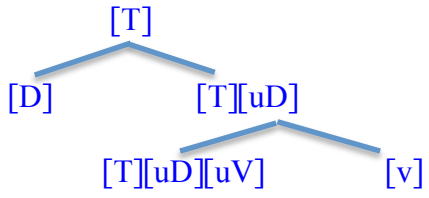
To illustrate this, take DPs. Determines select nominal complements and are thus assigned the features [D][uN]. Merger of a determiner and NP results in an element labelled [D] as in (4):

(4)



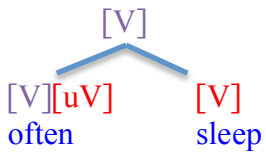
Predictions: This proposal predicts that the traditional view on labelling as 'projection by selection' (cf. Chomsky 1995) is essentially correct. If a particular head selects for a particular complement, the interpretable features of the selecting head will provide the label of the top node; and similarly, if the head has two selectional features it will first select for its complement, and second for its specifiers, and both selectional features, as well as the features of the head and complement will not percolate, resulting in the top node carrying the features of the head, as illustrated for the TP (where I take the ordering, first merger with v, second with D, to be conditioned by interface requirements):

(5)



Adjunction: A major advantage of this approach is that it does not only apply to the labelling of head-complement or specifier mergers, but also to adjunction. This is an advantage as virtually all other versions of bare phrase structure have no way of dealing with adjunction and need to allude to other mechanism (pair merge, late insertion, separating labelling from concatenation, cf. Hornstein & Nunes 2008 for discussion). The reason why adjunction is problematic for bare phrase structure is well known: why is it that adjunction to an XP has an XP label too. Why does adjunction to XP yield the same label XP, even though the lower XP is already a maximal projection? However, looking at the featural properties of adjuncts, things may actually follow naturally. One property of adjuncts is that they do have categorial status (e.g. being adverbs), but that they do not change the categorial status of the element they merge with: *often sleep* is categorially identical to *sleep*, a second property of adjuncts is that they are not selected, but select some modifiee themselves. The only way to encode that in this system is by assuming that adjuncts, like adverbs, have 'symmetric features': they carry an interpretable and an uninterpretable feature of the same kind. An adverb, such as *often*, carries [V] and [uV]. Adverbs are just verbs that select verbs. Now, let's see what happens if an adverb and a verb merge (see (6)). Both *sleep* and *often sleep* carry a feature [V]. But it is not *sleep*'s [V] feature that projects: it's the adverb's [V] feature (as indicated by the purple colour representing the projection line). Consequently, both *sleep* and *often sleep* have exactly the same featural properties (they are both maximal projections of [V]) and should therefore indeed have the same syntactic distribution.

(6)



Conclusion: The proposed labelling algorithm shows why the result of merge carries the features of its selecting daughter, in a straightforward way, and applies to both canonical cases of selection as well as adjunction.