A timing approach to escape hatch dependencies

Gautam Ottur (University of Göttingen)

Background. A common view of locality in syntax is that extraction out of various locality domains is disallowed, unless the extracted element is structurally at the *edge* of that locality domain. Locality effects like successive-cyclic movement (SCM) are thought to occur because everything except for the edge of a phase is transferred to the interfaces and becomes inaccessible to further operations.

But phases raise at least two questions: first, why should certain elements be phasal and not others, and second, why are phase edges exempt from Transfer? A recent counterproposal from Adger (2024) reduces locality to a general constraint on movement out of specifiers, rather than phases. In his account, he assumes that elements can have maximally two dependents (the first is its complement and the second its specifier, see also Kayne 1994; Brody 2000). Simplifying greatly, his proposal culminates in the following constraint: extraction from any specifier a is disallowed unless the extractee b is the specifier of a, i.e. the second dependent of a.

This can straightforwardly derive various CED effects, while allowing an 'escape hatch' for exceptions to the CED. Adger further stipulates that direct object positions are not complements of V, but rather specifiers of a higher functional category O. This makes a welcome prediction, as extraction from a CP-object (now a specifier) is then ruled out, unless the moving element first moves to Spec,C, thereby essentially deriving the CED and traditional SCM using a single rule.

Problems. But while Adger's proposal sidesteps some of the uncertainties of phases, his assumptions now fail to predict at least three cases which are empirically observed: extraction from an adjunct that has no specifier, extraction of multiple elements out of a specifier, and opaque specifier edges. I review examples of each of these below. First, assuming that only specifiers of specifiers can be extracted poses problems for PP adjuncts, which he treats as specifiers. In (1), Adger is forced to posit that there is some larger structure within the PP which allows the PP object to become a specifier of the adjoining category without violating anti-locality (Abels 2003).

(1) *What$_i$ did John cut the bread* [$_\text{P}$ *with* ___$_i$]*?*

Second, this kind of proposal would apparently also rule out iterative Ā- and A-movement across clauses, e.g. in hyperraising in Brazilian Portuguese (2), as multiple elements would need to merge as specifiers to C to escape the CP, which is itself presumably still the specifier of O.

(2) [*Quais livros*]$_i$ *elas$_j$ parecem* [$_\text{C}$ ___$_?$ *que* ___$_j$ *leram* ___$_i$]
which books they seem that read
'Which books do they seem to have read?' (Kobayashi 2020:18)

Third, it has been generalized that left-branch extraction (LBE) is licit only in languages without a definite DP layer in the nominal domain (Bošković 2005, *et seq.*). But LBE is also widely disallowed in languages without definite articles, such as Malayalam. Interestingly, Malayalam adnominals may Ā-move to the edge of NP/DP (3a). If the adnominal here is indeed at the edge, then it is expected to be extractable, but this is not borne out (3b).

(3) a. *rāmaṉ* [*bhaṃgiyuḷḷa$_i$ orŭ* ___$_i$ *vīṭŭ*] *vaṅṅi*
R. beautiful INDEF house bought
'Raman bought a beautiful house.'

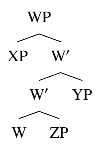
b. * *bhaṃgiyuḷḷa$_i$ rāmaṉ* [___$_i$ *orŭ* ___$_i$ *vīṭŭ*] *vaṅṅi*
beautiful R. INDEF house bought
'Raman bought a beautiful house.'

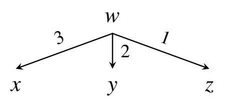
Proposal. To summarize, while phases encounter some conceptual issues, Adger (2024)'s alternative relies on distinctions between complements and specifiers to account for locality, which poses empirical problems. I will suggest an alternative approach which reformulates Adger's basic intuition, while allowing sufficient flexibility to account for the three kinds of cases above. Following Krivochen (2023), I will assume that Merge only takes atomic inputs, and generates dependencies between them (notated $\langle \alpha, \beta \rangle$), rather than a recursive set structure (4a). Each dependency is pushed to a stack, which is represented as a *graph* (4c) rather than a tree (4b). On these assumptions, merging a dependent with a root node critically does not extend that root node, because Merge does not create new nodes. Root nodes are simply nondependents with dependents of their own, and therefore can only be extended by them becoming dependents of other nodes.

(4) a. Set = {X, {Y, {Z, W}}}

c. Stack = $\big[\langle w, z\rangle, \langle w, y\rangle, \langle w, x\rangle\big]$

b.





This kind of approach construes hierarchy in terms of timing rather than constituency. Note also that 'workspaces' are not separated from each other here; all dependencies within a sentence are ordered relative to one another in a single stack. This requires that the dependencies that compose the subderivation of a specifier/adjunct are absolutely ordered in relation to those composing whatever they merge into. I propose that there is a general organizational principle like (5), which in effect prevents the root of a subderivation from being extended after its host structure is built.

(5) The extension of the root node of a subderivation D_1 cannot be interrupted by the extension of the root node of another subderivation D_2 and resumed later.

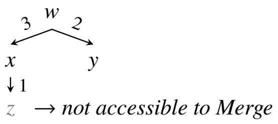
This necessitates that specifiers/adjuncts be built up to their root node as early as possible. Then, the opacity of specifiers/adjuncts can be derived from the following locality constraint:

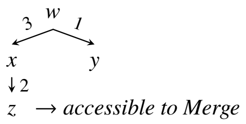
(6) Once a lower sister of a node α is found in a stack, any internal structure dominated by α found lower in the stack is inaccessible to Merge.

So if $\langle w, x\rangle$ immediately follows $\langle w, y\rangle$, y is a lower sister of x, and nodes dominated by x (e.g. z) are inaccessible to Merge (7a). Escapees from a specifier D_1 are only accessible if they merge with the root of D_1 in a case where the root of D_1 does not need to be extended, per (5), and the merge site of D_1 is fully ready, to avoid (6). We can thus reconfigure the escape hatch as follows: if $\langle x, z\rangle$ intervenes in the stack between $\langle w, y\rangle$ and $\langle w, x\rangle$, only then is z accessible to extraction (7b).

(7) a. Stack = $\big[\langle x, z\rangle, \langle w, y\rangle, \langle w, x\rangle\big]$

b. Stack = $\big[\langle w, y\rangle, \langle x, z\rangle, \langle w, x\rangle\big]$





Implications. The upshot is that these 'stragglers' escape through edges due to *when* they merge, and not because the edges of specifiers are otherwise unique. If we accept the stipulation of the functional category O, stragglers make the same basic prediction as Adger's locality constraint w.r.t. SCM, albeit in a different way: what allows extraction out of embedded CPs is not merging at the edge of CP, but the point at which that merger occurs. And crucially, the three empirical cases sketched above may now also be straightforwardly explained: (1) is predicted, because stragglers don't have to be specifiers, (2) is allowed, as there is no limit on the number of stragglers can undergo SCM, and (3) is explained, if the adnominal is simply not merged as a straggler.

The resulting question then is why escape hatches are not uniform across languages. The explanation is that whether these mergers can be delayed must be learned as part of a language's grammar. This may be surprising, but note that this is just a single diacritic, and more complex ordering systems are commonly assumed (e.g. Müller 2010, Georgi 2014, Merchant 2019, a.m.o.). Along these lines, the typological rarity of LBE is also expected because it is not only necessary that a language allow adnominals to merge at the edge of the nominal domain, as in (3), but also that the language has evolved an additional grammatical rule that allows the relevant element to merge as a straggler. Thus, LBE is predicted to be typologically exceptional, rather than normative.