LING 611 Spring 2014

Notes on Linear Correspondence Axiom

[From an Uriagereka monograph]
"...there are two rather different ways in which Kayne's L or any such mechanism ... could be grammatically implemented. One is that L determines the organization of phrase-markers from the very moment they are constructed, early in the derivation. The other, that L fixates precedence conditions among terminals only upon hitting the specific component of grammar that organizes speech... That timing has consequences: an implementation of L of the latter sort leaves the possibility open that phrase markers are actually not linearized in those components of grammar that do not feed phonetic representation. Kayne's instantiation of L was of the first sort, while Chomsky 1994 pursued the second line of reasoning."

[From Hornstein, Lasnik, and Uriagereka 2003 (2007). The dynamics of islands: Speculations on the locality of movement. *Linguistic Analysis* 33:149-175.]
"Uriagereka (1999) proposes a dynamic model of derivations by way of the multiple application of Spell-out, relating it to linearization in the sense of Kayne (1994). The intuition behind the Multiple Spell-out (MSO) program is that certain bits of structure are removed from the syntactic computation via SO before the derivation terminates. Spelled-out structure is inaccessible to context-sensitive operations. This poses at least two questions: (i) What is the characterization of relevant chunks, and (ii) What observables do these emergent structures correspond to?

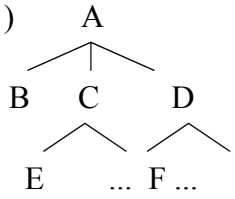
Consider (i). Uriagereka (1999) proposes that MSO can rationalize Kayne's (1994) LCA algorithm as follows: The standard version of the LCA... is conceptually problematic. It consists of two parts: a first rule that linearizes terminals that c-command one another [a], and second one that linearizes terminals that are not related by c-command [b].

Linear Correspondence Axiom (LCA): For A, B, terminals:
a. Base: If A c-commands B then A precedes B.
b. Induction: If C dominates A, and C precedes B, then A precedes B.

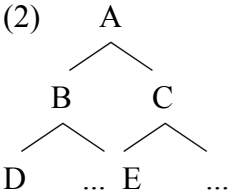
[From Islands grant proposal]
"The LCA base [a] is deducible from economy: C-command (if understood, as suggested by Epstein (1999), essentially as the history of Merge) is an emergent property of derivations; it is in the spirit of economy that this relation should be chosen as the linearization procedure, as any alternative would entail a more cumbersome method. That said, given derivational cascades naturally arise in 'command units' (where c-command holds as the only relation among terminals A and B). The LCA induction [b] is, however, unnatural. Eliminating it, the only direct way to linearize A and B is within a command unit. Spell-out comes to the rescue of any A that does not c-command B, literally flattening the structure C that does c-command B and also dominates A. This turns C into a terminal, thus its component parts (including A) are no longer active syntactic constituents. The reason A is linearized with respect to B, then, is no different from why the units within a compound precede whatever the compound precedes. Spell-out applies only when necessary for convergence (here, linearization), simply because economy prevents it from applying unnecessarily."

Complex Branching and Linearization

(1)

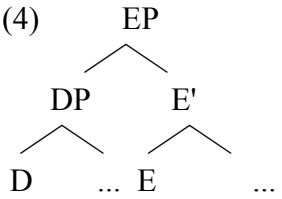


(2)



(3) Linearization Procedure for terminal elements (Kayne's LCA interpreted as a spell out procedure á la Chomsky)
a. BASE: If X asymmetrically c-commands Y, X precedes Y.
b. INDUCTION: If X is dominated by Z, and Z precedes Y, X precedes Y.

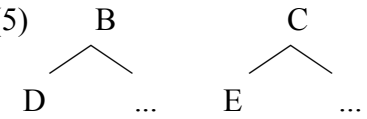
(4)



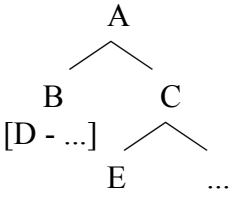
Where E and D are arbitrary heads
DP and EP are maximal projections
and E' is an intermediate projection

Dynamic Spell-out? Uriagereka (1999)

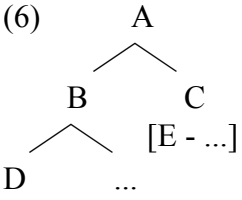
(5)



'Flatten out' B to merge it with C.



(6)



(7) Linearization Procedure for terminal elements
X precedes Y iff X asymmetrically c-c-commands Y.

(8) This is simpler than the original. AND it explains why specifiers (and maybe adjuncts) are islands.

[From HLU (2003/2007]

Islands and Repair-by-Deletion

Ross (1969) was the first to observe island repair under Sluicing as in (9):

(9) That he'll hire someone is possible, but I won't divulge who (*that he'll hire is possible) [Sentential Subject Constraint]

To account for such phenomena, Chomsky (1972) suggests that diacritic '*' is assigned to an island when it is crossed by a movement operation; an output condition forbidding '*' in overt structures accounts for the deviance of island violations. But if a later operation (Sluicing in this case) deletes a category containing the *-marked item, the derivation is salvaged. Merchant (2001) and Lasnik (2001) develop this kind of analysis, whereby islands constrain movement by creating what are ultimately 'surface' defective structures -specifically PF defective- in terms of recent transformational theorizing.

Our MSO approach, outlined above, provides a reasonable understanding of the '*' diacritic and its elimination. In fact, a structure as in (9) is unlinearizable. We actually must produce an unlinarizable object in order for the computational system to reach into the boldfaced sentential subject, to extract who. If (9) were attempting to surface as it is, with the unlinearized subject, it obviously couldn't make it into PF. However, PF deletion rescues the example: It turns off the parallel PF material that includes the offending unlinearized chunk, thus allowing its representation to make it to LF:

(10) That he'll hire someone is possible, but I won't divulge who (*that he'll hire t is possible)

At the point where a spec or adjunct is merged, there is a choice of whether to linearize it or not. If it is linearized, nothing can be extracted. If it isn't, extraction is possible. For reasons of cylicity, there is no later opportunity to linearize. Hence linearization will ultimately fail, unless all the problematic material is gone.