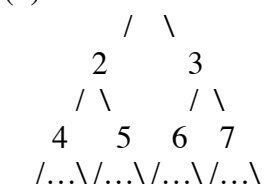
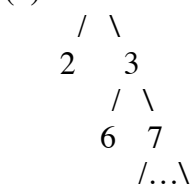
Multiple Spell-out: *Linearization Matters*

(1) *The FS Limit on Phrase Structure*
An exhaustively binary phrase-marker, none of whose branches symmetrically bifurcates, can be expressed in a FS fashion.

(2)


(3)


(3) [1] → [3] → [7] → […]
2 *6* …

(4) When *x* asymmetrically c-commands *y*, *x* is linearly ordered with regards to *y*.

(5) If *x* is linearly ordered with regards to *y*, then *x* precedes *y*.

(6) If *x* is linearly ordered with regards to *y*, then *x* follows *y*.

Many other possibilities come to mind: in relevant conditions *x* could be *k* steps removed from (following or preceding) *y*, where *k* could be calculated in any number of ways. For any number *n* of symbols organized in a string, *n!* linearizations are viable.

(7) *Linear Correspondence Axiom* (LCA, partial statement)
When *x* asymmetrically c-commands *y*, *x* precedes *y*.

(8) *Mirror Linear Correspondence Axiom* (MLCA)
When *x* asymmetrically c-commands *y*, *x* follows *y*.

(9) […] → [7] → [3] → [1]
… *6* *2*

For a sentence like *he is biting them*, we would obtain a PF along the lines of *them biting is he*. This is pronounceable, so why is it not the way the language faculty chose?

(10) *Characteristics of (long) Context-sensitive Dependencies*
a. PF: Collapse at a unique *pronunciation occurrence*.
b. LF: Collapse at separate *scope* and *variable* interpretations at different occurrences (interpretation of *restriction* at various 'reconstruction' sites).
c. Syntax: Satisfaction of *Last Resort*, *Locality* and *Uniformity* conditions.

(11) a. PF: *Which man did Fido bite Ø*
b. LF: [Which [Fido bite man]]
c. Syntax: i. [Fido bite which man]
ii. [which man (did) [Fido bite which man]]

Grammars represent this context-sensitive dependency by having **the operator asymmetrically c-command its variable.** Now could things have been the other way around? Not under standard conditions of strict compositionality. Consider:

(12) [man [Fido bite which]]
(13) [which man (did) [Fido bite which man]]

Ackema and Neeleman 2002 argues that:

(14) Parsing considerations favor the spec-first option (implicit in the LCA), *if specifiers involve moved items*.

Right away it should be pointed out that not all specifiers seem to involve movement:

(15) *In those days* [***there*** [*used to arrive* ***trains*** *full of refuges*]] (, *didn't there?*)
(16) *I wonder* [***whether*** [*the universe is really still expanding*]]

The reasoning is based on two assumptions: (i) that **'already parsed structure' is removed from the parse** to reduce on-line memory loads and (ii) that **parsers are 'gap-searchers',** once relevant displaced antecedents are recognized. Assumption (i) can be granted (though more below). Assumption (ii) is based on the intuitive fact that, **due to their being silent, gaps are hard for the parser to find,** while antecedents are generally more apparent, being phrases with a phonetic representation and often even an associated cue (e.g. a given complementizer shape). But is it *necessary* that antecedents should be reliable anchors for long-distance dependencies, while corresponding gaps should not?

Matters seem tricky in two respects. First, **many languages signal major gaps through phonetic cues,** for instance agreement markers as in Basque:

(17) a. *Nork esan duzu bidali diola liburua Mariari?*
Who-S said III-AUX-II send III-AUX-III-III-C book-the-O Mari-IO
'Who have you said sent the book to Mary?'
b. *Nori esan duzu bidali diola Jonek liburua?*
Who-IO said III-AUX-II send III-AUX-III-III-C Jon-S book-the-O
'Whom have you said Jon sent the book to?'
c. *Zer esan duzu bidali diola Jonek Mariari*
What-O said III-AUX-II send III-AUX-III-III Jon-S Mari-IO
'What have you said Jon sent to Mary?'

In this language all arguments agree with a verbal form, marked with roman numerals in (17). **Gaps associated to Wh- antecedents are thus clearly located** (antecedents too are

overtly marked with case forms, glossed in (17) as S for subject, O for object and IO for indirect object). In other languages, such as Lebanese Arabic, gaps are signaled through resumptive clitics (data from Aoun and Li 2003):

(18) a. *?ayya mmasil Seft-uu be-l-mat?am*
which actor saw.2sg-him in-the-restaurant
'which actor did you see in the restaurant?'
b. *miin ?enbasatto la?inno saami ?arraf-o ?a-miin*
who pleased.2pl because Sami introduced-him to-whom
'who were you pleased because Sami introduced (him) to whom'

For these sorts of 'gaps', it is unclear why there should be any parsing issue, any more than there is in finding a corresponding antecedent/postcedent. If both dependents in the long-distance relation are clearly signaled, the only genuine concern should be to detect the dependency itself, and at that point, **why should it be easier to spot starting from one end than from the other.** Note that **both Basque and Lebanese Arabic displace Wh-phrases to the front of sentences,** like all other known spoken languages.

Moreover, **not all antecedents are easy to spot, vis-à-vis corresponding gaps.** Consider displacement in a pro-drop language like Spanish:

(19) a. [*Tú y yo*] *parece**mos** haber sido elegid**as***.
you and I seem-I.pl to.have been chosen-fem.-pl.
'You and I seem to have been chosen' (said by a female).
b. pro *parece**mos** haber sido elegid**as***.

In (19a) a grammatical form is needed to trigger the agreement in the passive form and in the matrix verb ('1st person plural' is not a default form of agreement in Spanish). The standard way to achieve the relevant dependencies is by generating **the grammatical subject as the logical object of the passive,** as in (20a), thereby displacing this element to the front of the sentence, possibly in successive steps as in (20b):

(20) a. *parecemos haber sido elegidas* [*tú y yo*]
b. [[Tú y yo]$_i$ / *pro*$_i$ parece**mos** [t_i haber sido [t_i elegid**as** t_i]]]

Now in the version of the sentence in (19b), **the 'antecedent' is *pro*, which as (20b) shows is the silent antecedent of a trace (or several).** It may be argued that the agreement form on the main clause in these examples (*parece**mos***) is a sufficient phonetic cue to parse the null antecedent; but **not even this agreement form is necessary in other instances,** for example (21):

(21) pro$_i$ *parecer* [t_i *haber sido* [t_i *elegid**as*** t_i]]] *fue un honor (para **nosotras**)*
to-seem to-have been chosen was an honor for us
'To seem to have been chosen was an honor (for us).'

In fact, if anything in a sentence like (21) what seems easier to identify is the gap (associated, via overt agreement, to the passive *elegid**as***) not the silent antecedent.

Consider a **hypothetical parser that were concerned with 'postcedent searching',** for dependencies turned around from standardly anaphoric to relevantly cataphoric (in the sense of a clearly identified 'gap' for a possibly null postcedent). In other words, **a parser responsible for encountering PF forms as in (22a), for the LF in (22b) and the (internal) syntax in (22c):**

(22) a. PF: $_{\text{RESUMPTIVE}}$ *bite Fido did (man which)/Ø*
b. LF: [Which [Fido bite man]] [linear order irrelevant]
c. Syntax: i. [Fido bite which man]
ii. [which man (did) [Fido bite which man]]

That sort of a parser should also be concerned with the **efficient resolution of the putative long-distance dependency** as an 'antecedent searching' parser is, **to reduce pressure on parsing resources and augmenting parse reliability.** But matters would be backwards: for a null postcedent (meaning *which man*) following its overt (resumptive) gap, that postcedent could in principle be found in any subsequent domain.

It is surprising that, although there are ample instances of signaled antecedents as in (17)/(19), and even null antecedents as in (20) or (21), **the sort of situation in (22a) seems very uncommon.** We do see an effective instance of that in (21), but these are **normally local domains (of so-called A-movement)—and cf. also Backwards Control.** What seems **harder to find is a version of (22a) involving Wh-movement, even when it would make sense.** For instance the Spanish (23b), alongside (23a):

(23) a. *El libro, el cual todos quieren leer, ha llegado.*
the book the which all want.III to.read has arrived
'The book, which everyone wants to read, has arrived.'
b. pro *(*(el cual) todos quieren leer(lo)), ha llegado (; el libro, digo).*
the which all want.III to.read-it has arrived the book say.I
'(It) has arrived; the book, I mean.'

The question is **why the relative clause cannot be headed by a null antecedent, on analogy with 'head-internal' relatives** in Korean (data from Chung & Kim 2003):

(24) a. Externally Headed Relative Clause:
John-i [$_{\text{RC}}$ *Op$_i$ Mary-ka e$_i$ kwup-ø-un*] *kamca$_i$-lul mekessta.* [Korean]
John-Nom Mary-Nom bake-prf-RL potato-Acc ate.
"John ate the potato Mary baked."
b. Internally Headed Relative Caluse:
John-i [$_{\text{RC}}$ *Mary-ka kamca$_i$-lul kwup-ø-un*] *kes$_i$-ul mekessta.* [Korean]
John-Nom Mary-Nom potato-Acc bake-prf-RL kes-Acc ate.
"John ate the potato, which Mary baked."

In other words, why can't (23b) have the import of something like **'it, everyone wants to read it, has arrived, that book'.** Of course, for that to be possible we would have to effectively allow a **totally null Wh-operator**; not just null in the sense of expressions like 'there's the woman (*who*/*that*) I love', but rather **an expression completely devoid of PF representation, directly or indirectly.** That seems out of the question, suggesting that there is something real to the idea that, **when long-distance dependencies are formed, they surface in anaphoric terms, not cataphoric ones.** The issue is why.

One thing seems clear: **When finding context-sensitive dependencies, relevant parsers have to be alerted to a complex operation involving serious memory resources.** As, for example, Kobele 2006 shows, a parser of an LCA-based grammar can put the operator on a memory buffer, until it finds the appropriate variable; in contrast, **a hypothetical parser of a MLCA-based grammar must put *the variable* instead on its memory buffer, until it finds the operator.** Operators and variables may stand in a binding relation (the normal scenario), or not. If not, the operator operates vacuously and the variable is free, both straightforward possibilities within **logic.** In grammar, however, conditions are narrower (setting aside parsing concerns): **free variables are viable *but not vacuous operators*,** as they violate the Principle of Full Interpretation (Chomsky 1986a). In the end, under conditions of full interpretation, all grammatical tokens are either eliminated prior to interpretation (if uninterpretable, for instance pleonastics or case-markers) or else interpreted. In contrast, a free variable can receive an interpretation either through existential closure or unselective binding. This is because **the variable can be set in extra-sentential terms,** invoking discourse representations or similar devices.

An interesting asymmetry:	*Logical Space*	*Grammatical Space*
Variables	ASSOCIATED √ FREE √	ASSOCIATED √ FREE √
Operators	ASSOCIATED √ FREE √	ASSOCIATED √ **FREE ***

Operators must be associated to a variable, or they are not interpreted; variables can be grammatically associated (bound) or not. **Consider the effect this has on a parser.**

An LCA-based parser that encounters an operator Ω(*x*) is in grammatical need of variable *x*. Its computational demand is what it is (due to the operator-variable dependency), but resolvable within the confines of derivational memory specifications necessary to scan the operator-variable search space (see Berwick and Weinberg 1984). In contrast, **an MCLA-based parser that encountered a variable *x* would be in an equivocal territory: a priori *x* may be a variable to which an operator Ω(*x*) is associated, or a free *x*. The system won't be able to resolve this issue until the relevant search space where *either option* can be satisfied is completed.** If Ω(*x*) is in fact found within that search space, then the computational demands for this parser will turn out to be no different from those in a standard parser which has processed an operator and waits for a variable. However, if no such operator is found – or, rather, *while said operator is not found* – the system cannot make any decision on how to interpret this variable. Importantly for the examples we saw, **this doesn't depend on the phonetic representation of operators or variables: it is a grammatical status cued to their very nature as symbols.**

Baader and Frazier 2004 shows how these matters affect even regular parsers, in **situations where variables could be equivocally interpreted as free or bound,** reducing parsing speed. In a MLCA-based parser that difficulty would be the normal state of affairs, drastically reducing parsing ease, at least as compared to a corresponding LCA-based parser. So the present account of why specifiers ought to be first is based on the grammaticalization of a parsing fact. The reason, however, that the LCA-based parser is considered more efficient is not based on the gap-searching capacity of the parser, but rather on its ***variable*-searching, whether these are gaps or, instead, phonetically represented forms.** From the present perspective, parsing considerations still favor the LCA over the MLCA. **Hitting an operator first is a direct signal of a context-sensitive relation. In contrast, hitting a variable first is not a reliable signal:** there could be an operator coming, but then again there may not be one at all; holding the variable in memory waiting for the operator-that-never-arrives is a waste of resources. Seen that way, the explanation also extends to the factually problematic examples in (16), which involve no antecedent/gap relations. *Whether* is an operator; so although it hasn't moved or left a gap, its occupying a left-ward specifier is expected under the present view.

Consider next those **chains whose context-sensitive relation expresses an Agree relation between an unvalued Probe and a valuating Goal** (Chomsky 2000), or devices of similar formal complexity (to be compared to the situation in (12)):

(25) a. PF: *A man was bit Ø*
b. LF: [was bit [a man]]
c. Syntax: i. [**was** [bit [**a** man]]]
ii. [[a man] was [bit [a man]]]

Our main concern here is whether the necessary step in (25c, i) creates troubles in the mirror linearization. The **valuation implied in (25c) necessitates c-command,** as much as scope does. In context-sensitive Agree dependencies, a feature F of the Probe is an *unvalued attribute*, which seeks an appropriate value V under the following conditions:

(26) *Conditions on Value Search from Probe P to Goal G* [PARTIAL]
a. G is in P's complement domain.
b. G is accessible to computation.
c. G is local to P.
d. VALUATION: A feature υ-Φ within G constitutes a valuator for attribute Φ within P [if and only if?] Φ in G is identical to Φ in P.

When a value υ for Φ within P is located in G in the conditions in (26d), υ-Φ from G is taken to be syntactically expressed in P. This entire valuation mechanism goes **outwards in the phrase-marker,** from the moment the search starts (26a) to the moment feature valuation is implemented, which is what creates the context-sensitive formal object whose derivational origin is at G but whose final syntactic expression is at P. A comparable **valuation could only go inwards** into the syntactic object (thus obeying some form of anti-c-command) **if derivations proceeded 'root first'.**

Remember: **the MLCA also involves c-command relations; it just maps them to PF in the opposite direction as the LCA does.** Now: the parsing argument built for operator-variable relations can be reproduced for the more basic valuation processes just studied. Compare two sorts of situations that arise in many languages:

(27) *Ways of valuating*

a. **Fueron** mordidos [**varios** hombres].
Were.III.PL bit several.III.PL men
'Several men were bit.'

b. Hay [**varios** hombres] mordidos.
Have.III.SG.loc several.III.PL men bit
'There's several men bit'.

In (27a) Agree has taken place without an ensuing Pied-Piping. The Probe is the boldfaced ***fueron*** 'were' and the Goal is ***varios*** 'several'. In (27b) the only putative Probe for the goal ***varios*** would be the existential *hay* (or possibly a null pleonastic in its specifier), which carries the agreement features that are presumably unvalued. In this instance, however, the matching in (26d) is not obviously met. If so, given the mechanics in (26) two conclusions: the valued feature in the Goal ***varios*** can (unsurprisingly) survive the derivation without being displaced to the Probe site, and the unvalued feature in the Probe ***hay*** (more surprisingly) gets valued or else erases from computation, however this is achieved. Apparently, **such processes are possible only with *default* feature values,** such as 'third person singular' (although default is language-specific).

So when facing a Goal, two possibilities emerge, depending on Probe specifications: **either a full match,** along the lines in (26), or **else a default implementation** as in (27b) ensues. Parsing-wise, **the goal is not good at telling which of the options to take,** because the decision doesn't depend on it: a Goal can never prevent a Probe from going into the default strategy. In contrast**, the Probe plays with all its cards: if it is unvalued, it will go into the Agree process; if valued, it must be so by default.** The parser encountering such a Probe doesn't need to wait to see which strategy is operative. Finding anything but default values in a T probe (all relevant combinations of ϕ-features) entails having to establish an Agree relation; if, instead, default values are encountered, two possibilities emerge: either the default strategy or, possibly, an Agree with a Goal which happens to have values that coincide with the default ones. In contrast, **a parser that encountered goals systematically first – like the MLCA-based parser – would constantly be in this equivocal scenario,** not being able to resolve it until corresponding Probes are found. The approach extends to an explanation for why (15) involves a leftward specifier even if not involving displacement.

That version of the argument, albeit more technical and dependent on minimalist instantiations of context-sensitivity, is **arguably deeper.** The **operator-variable relations examined may actually be implemented in terms of some generalization over the valuation processes** just sketched. This may allow for a unification.

To be fair with the MLCA, one could ask **why the entire parser couldn't work backwards**. Although logically possible, **this alternative wouldn't make sense from the point of view of memory,** understood as the ability to carry the flow of information *forward* in time (Gallistel & King 2009). If memory worked backwards, it would make sense for parsers to work from after to before. The specific parsing argument given above is based on **a) the equivocal interpretation of variables or Goals, vis-à-vis corresponding operators or Probes,** which are interpreted univocally and **b) the fact that the parser seeks to make fast univocal decisions in terms of what it parses early,** and does so linearly and incrementally. **(a) is a point about the LF side of grammar, in the case of operator-variable dependencies, and about syntax in the case of Agree relations** (or perhaps about syntax in both instances, if a unified account can be developed). In contrast **(b) is a point, at least in part, about PF, where 'early' and 'late' make simple linear sense.** For a telepathic being there may still be 'early' and 'late' matters, for instance organizing thought in even normal (non-telepathic) creatures. Whether that, too, may independently encourage an LCA-parser in detriment of an MLCA-parser, however, seems impossible to determine with our present understanding of brains (particularly if they can run in multidimensional ways). **Only in speech are we certain of a before and an after, locally and long-distance.** It is **only through the conditions that speech imposes that we can univocally decide for an LCA grammar.**

Within a minimalist system, this suggests that **Chomsky's (1994) interpretation of Kayne's L is more straightforward than Kayne's in (1994).** For Chomsky L is nothing but a way of deciding on the Squeezing Problem, at the point where this matters: while mapping syntax to PF; for Kayne, instead, L is a defining characteristic of phrase-markers in general. The fact that L in his specific LCA terms is what the grammar chose for context-sensitive dependencies, as opposed to the a priori equally reasonable MLCA, suggests that **the grammar is sensitive to interpreting operator-variable relations – or Agree relations in general – in a way that reliably anchors their univocal representation to expressible time.** The linear nature of PF, in a biological system that can control it with the precision that we can, suggests that PF became, in evolution, something like our *usable clock* of before and after. If humans hadn't had this reliable 'PF clock', we may not have been able to opt for the LCA as opposed to the MLCA – particularly because, for a simple class of structures, the MLCA is actually more natural.

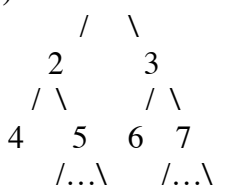
For consider the bottom-up syntactic architecture we are assuming. The derivation starts with a chunk X and some other chunk Y is added; then Z and so on. **Operationally we go from X to Y to Z, a version of the FS situations** discussed. And what is more natural for the object in (28a) to map as, in phonetic terms: **(29b) or (29c)?**

(28) a. X → Y → Z
b. <X, Y, Z>
c. <Z, Y, X>

It is in fact **possible that the MLCA procedure is, in some form at least, still operative in the language faculty,** albeit in a reduced set of circumstances.

Obviously human sentences are more complex that what the FS limits imposes. **Structures that do not fit the FS Limit should not be linearizable either by the LCA or the MLCA, as in those structures it is not true that any given terminal symbol *x* will stand in an asymmetric c-command relation with any other terminal *y*.** And yet the Squeezing Problem will only be fully addressed if *all* terminals in a phrase-marker can be linearized. Kayne resolves this matter by **complicating the LCA.** In effect, he chunks the problem down by **allowing the linearization not just of terminals, but also of non-terminal symbols.** The relevant situation is as in (29), the problem arising, say, for the linearization of *4* and *6*, for which c-command relations cannot be established:

(29)



In Kayne's system *4* linearizes before *6 indirectly*, essentially because we think of *2* as a terminal, linearizable with respect to *6*. Then **a proviso is added** to the following effect:

(30) *Linearization Induction*
If a non-terminal X dominates a terminal *y*, and X is linearized with regards to terminal *z*, then *y* is linearized with regards to *z*.

If *2* linearizes with respect to *6* in (29), then *4*, which *2* dominates, will too.

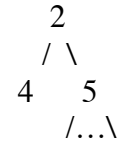
If attempting to deduce L from the Squeezing Problem, that sort of solution seems dubious. After all, **if we are attempting to list *terminals*** (PF being about the speech characteristics of such elements), **how come the induction mechanism in (30) makes reference to non-terminals?** Why didn't the language faculty just 'gave up', declaring the relevant structures too complex to linearize? This is the reasoning Kayne used in (1994) to rule out ternary branching. Effability demands alone do not make a coherent thought expressible in linguistic terms, so the grammar could have succumbed when facing (29). What we need to understand is **what trick worked this time, but it did not for ternary branching.**

To make things worse, once we're willing to establish the claim in (30), there is **no easy way of stopping it from overgeneralizing wildly**. Thus, observe that in (29) **non-terminal *3* asymmetrically c-commands *2*, hence should linearize with regards to *2*,** and then as per (30), since *3* dominates *6*, *6* should linearize with respect to *2*; but we concluded that *2* linearizes with respect to *6*. Both **Kayne (1994) and Chomsky (1995) have *ad hoc* ways of avoiding this paradox,** by making one of these conclusions not count, so that the contradictory one prevails. However, **their (different) stipulations are impossible to motivate beyond the purpose for which they are postulated;** the opposite stipulation would seem equally natural, though yielding the wrong facts.

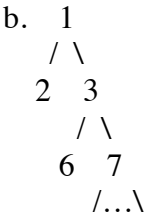
Having a ***single*** **application of the rule of Spell-out to split the derivation was a residue of S-structure.** The possibility was thus raised of, under certain circumstances, encountering situations of *multiple* Spell-out (MSO). **Economy considerations alone rule out rampant MSO,** under the assumption that a single rule application is preferred over multiple ones. However, **economy considerations apply only *up to convergence*.** If in order for a derivation to converge it needs to apply a given rule several times, and applying that rule a lesser number of times leads to a non-convergent derivation, then there is no valid competition between those alternatives. Only the convergent one counts. That situation obtains for structures along the lines in (29). **Suppose only the LCA as in (8) exists in the grammar – not the extension in (30). Then (29) couldn't converge, if L in terms of the LCA applies at a single point of Spell-out.** However, what if we break (29) into linearizable chunks, *each obeying the FS Limit for which the LCA works*? Barring stipulation, nothing prevents that. To start with, because **it addresses the symmetry difficulty that neither Kayne nor Chomsky could resolve without stipulation.** Note that what's being said is that (29) should, first, be divided as in (31):

(31)

a.



b.

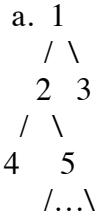


Then **within each of these phrase-markers** the LCA as in (8) applies, correctly. Finally the **top symbol in (31a) and the corresponding one in (31b) are made to correspond.**

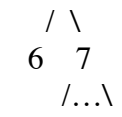
The **contradictory linearizations issue never arises, because for that to be possible we must have chunked (29) as in (32):**

(32)

a.



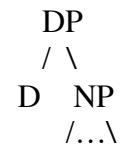
b.



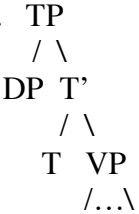
As such, this is a straightforward (and still lethal) possibility. But now **observe what happens when we replace the numbers in (31) or (32) by familiar symbols:**

(33)

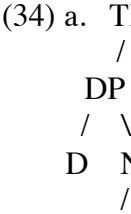
a.



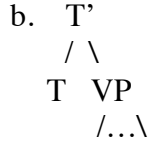
b.



(34) a.



b.



The tree-division in (33) ***doesn't tamper with any lexical projection*****; in contrast, the tree division in (34) does.** This difference is in the structures, bearing in mind:

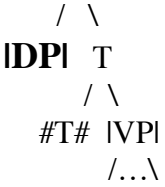
(34) No Tampering (Chomsky 2000) [No overwriting – Uriagereka 1998]
The computational system does not alter labeling mechanisms, as labeling procedures are incrementally tracked and monotonically built, with no back-tracking. Suppose: not even "bar-level" changes are allowed.

The situation in (34) can be excluded **if the 'bar' notations in these phrase-markers are substituted by 'bare' phrase-structure notions** (where '#' signals a lexical item and 'I' a maximal projection—and elements can be both; no projection symbol around a non-terminal signals an intermediate projection):

(35)

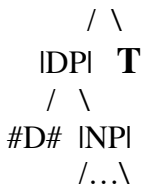
a.

b.

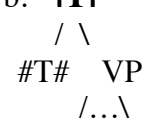


(36)

a.



b.



In (36) **T in (a) and |T| in (b) do not match,** and therefore the overall tree in this instance cannot be recovered: **T' in (36b) comes out as a maximal projection because** ***in that phrase-marker*** **it is not immediately dominated by any other category of the same lexical type**. In contrast in (35) this difficulty doesn't arise. In particular, **|DP| in (a) and (b) is the same sort of category (a maximal projection in each instance) because we haven't tampered with its projection in the tree division, and thus at no point does |DP| become a new sort of projection.** This will happen whenever we divide the phrase-marker in such a way that both divided chunks result in a maximal projection. That is **close in spirit to Chomsky's (1995) stipulation (which prevents the computational system to have access to intermediate projections); however, we need not say anything specific about intermediate projections.**

Although now **the issue is one of** ***matching the separate chunks*** **in the divided tree,** a natural and independent (*and* extremely non-trivial) demand for any system sending bits of structure to interpretation in different derivational cycles.

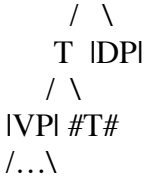
(37) *The Address Issue*

Whenever a phrase-marker K is divided into sub-components L and M for K to meet LCA conditions of multiple Spell-out, the daughter phrase-marker M must correspond to an identical term M in L.

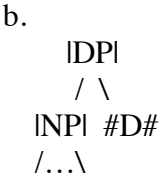
There are **two (more and less radical) ways of executing the Address Issue.** What is of some interest at this point is the architectural demand in (37) in itself, the fact that it presupposes a ***correspondence mechanism*****, thus an issue of systemic memory.**

It is worth asking **what would have happened to the system if the Address Issue had to be stated in MLCA terms,** instead of the LCA terms assumed in (37). The situation arises for objects as in (35), which in MLCA terms would come out as in (38):

(38) a.



b.



In MLCA terms, the grammar would need to spell out a complex phrasal object in such a way that: (i) upon hitting a specifier, this element comes in last, and (ii) if its structure is phrasally complex (DP in (38)), it is stored in memory for further computation, while computation proceeds at the root of the structure (TP in (38)). So: **Is it better to store something 'as soon as possible' in the speech stream or, rather, 'as late as possible'?**

The matter can be clarified by replacing the abstract structures in (35) and (38) with actual words. Effectively, the two situations are as in (39):

(39) a. *The man ate meat* parsed as [The man]… [he ate meat] LCA PARSING
b. *Meat ate man the* parsed as [Meat ate he]… [man the] MLCA PARSING

There is a significant difference between each of these parsings. **What the LCA first hypothesizes is not a sentence, but what the MLCA parser does is.** Given the structure of how the information is presented in each instance, **an LCA parser cannot decide on a propositional interpretation with its incoming complex materials until it hits the rest of the structure,** where they belong as arguments. In contrast, **an MLCA parser, which is working with the very flow of the bottom-up activation, could make such a decision from the moment it hits the right verb, particularly if it shows agreement.** The issue is whether a fast decision here is better than a slower one. A different way of putting this is in terms of the parser being warned that a complex structure is coming. **The LCA parser may get a signal of the complexity of the specifier by encountering it first. In contrast, the MLCA parser does not 'know', until finally finding the specifier, whether it was in fact simple or complex.** Is it important for the parser to actually predict that it will be hitting a genuinely branching structure that it needs to

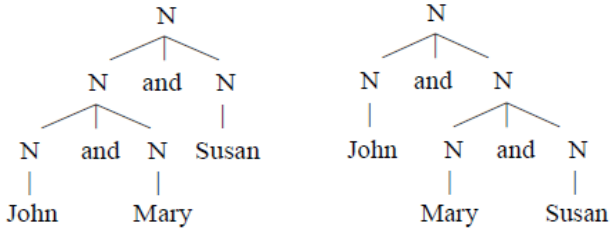
integrate in presumably non-trivial ways, or can it proceed assuming it is actually getting a structure of an effective FS complexity, later on to find out that more was to come?

Suppose that the parser had **two different modes, an FS mode and a Push-down (PD) mode**—in the latter instance activating **a stack** of some sort. Such a possibility is suggested by the different sorts of structures that emerge in relation to simple conjunction, as discussed in Lasnik and Uriagereka (forthcoming):

(40) [John and Mary and Susan] criticized each other.

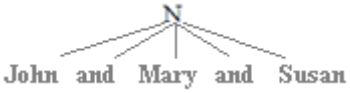
Structurally different parses are obvious for this structure, as in (42):

(41) a. b.



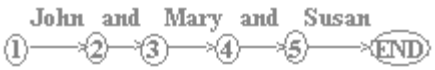
Corresponding semantics for these structures are also straightforward. One situation making (41) true involves John and Mary criticizing Susan, and vice-versa; that grouping of the event participants is naturally expressed as in (42a). But (41) would also be true if John criticized Mary and Susan, and vice-versa; and that grouping of the participants is naturally expressed as in (42b). However, one other way in which (41) comes out true is if *each of the three participants criticized each of the others*. Lasnik and Uriagereka argue that the most natural structure to correspond to such a semantics is flat:

(42)



Now as we saw in section 2 above, (43) is indistinguishable from (44):

(43)



Lasnik and Uriagereka propose that the grammar accepts fragments of the sort in (44). If so, a mere FS parser ought to process elements like (44). Differently put, **arguably the parser deals with (44) by ignoring its phrasal stack.**

It should be clear that **FS conditions can only be activated in instances in which these conditions are called for,** within the general PD specifications that obtain otherwise—or the structuring in (42) would never be viable. It turns out to be tricky to determine what the exact conditions are in which the FS activation is possible. Samuels 2008, following Raimy and Idsardi 1997, argues that this is the general format of

phonological representations. When it comes to syntactic representations, **list intonation helps signal an FS status, as do explicit conjunctive elements.** In other instances, as discussed in Uriagereka 2008: chapter 6, **iteration signals the FS parse, as in expressions like '*never, never, never surrender!*'** The parsing of an entire expression into a proposition also helps, which may be behind the paratactic analysis of certain embedded clauses that are thereby treated effectively as main clauses.

Under these circumstances, **a parser for a grammar obeying the MLCA would arguably be 'fooled' by structures as in (39b).** Such a parser would assume – until encountering the more complex structure – that said structures are of the simpler, FS sort, which this parser could process as something like a simplex *he ate meat*. Note that this parse would be legitimate, as noted, if in fact parsing an entire sentence/proposition is compatible with activating FS conditions, the general situation obtaining for parataxis. In contrast, **a parser for a grammar obeying the LCA would have to assume it is dealing with a complex structure right from the beginning,** since it may not reasonably assign a FS parse for the fragment it receives, a complex subject. So in a sense **the MLCA parser would garden path in these instances, assigning a flat structure even when it must deal with a non-flat structure,** given what the parser will eventually process (again, a complex subject). The LCA parser, in contrast, does not garden path: it assumes the more complex structure the moment it hits the complex subject; that is, early on.

The issue is worth pondering in principle, for it **may have also played a role in the decision between the LCA and the MLCA linearization procedures.** In this instance, instead of evaluating the asymmetries among elements within context-sensitive dependencies, the issue would rather be **the very complexity of phrasal expressions, and whether being fully recursive (complex on both sides) or not affects their parsing status.** The **LCA parser favors complex phrases being rapidly parsed as fragments of a structure which, at that point, is immediately recognized as fully recursive.** In contrast, a hypothetical MLCA parser favors a system whereby complex phrases may, on first pass, be taken to present mere tail recursion, not full recursion.

Actually *both* forms of parsing may well be necessary, particularly if the MLCA parser corresponds merely to FS fragments and the LCA parser corresponds to full PD fragments. Again, **a PD parser is an FS parser with a stack, so having a stack need not mean that it must be used,** any more than having an engine in a plane entails that the plain cannot turn the engine off and glide. It is an empirical question whether the human parser ever 'turns off its memory', but it is hard to understand what it means, otherwise, to parse the bracketed structure in (41) as (43). If it is indeed the case that the parser has FS and PD (FS + stack) modes, it is **interesting to suppose that the FS mode works in MLCA terms, while the PD mode works in LCA terms.** If nothing else, this would address recalcitrant situations of the form in (45), which are well-known potential nightmares for any LCA treatment, in that they involve 'rightward' dependencies:

(44) a. *I heard today a particularly beautiful story about immigrants*.
b. *John built a house rapidly cheaply, he didn't cheaply rapidly*.
c. *John saw m naked, drunk; he didn't see me drunk, naked*.