WINFRIED LECHNER

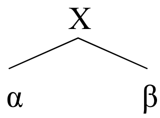
PHRASE STRUCTURE AND LINEAR ORDER

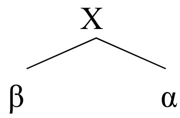
Webpage: http://vivaldi.sfs.nphil.uni-tuebingen.de/%7Ennsle01/HS2006.htm

---

1. STRUCTURE VS. PRECEDENCE

THE ISSUE: Syntactic principles are (for heuristic reasons) defined in such a way that they refer to structure (dominance and c-command), but not to order. If α is e.g. in a designated relation with β - say, Binding - then it does not matter whether α precedes β, or v.v.:

(1) a.


b.


This can be called the Dominance Hypothesis:

(2) **DOMINANCE HYPOTHESIS**
Syntactic principles refer to c-command/dominance, but not to precedence relations.

The tree can be likened to a mobile, whose root X is fixed, but whose branches swing freely in a two dimensional plane (see e.g. exposition in Uriagereka 1999). But at least at the point at which the product of the completed derivation is submitted to the PF component, an ordering of the two terms has to be specified. That is, the two dimensional tree has to be mapped onto a one-dimensional phonetic representation. More precisely, it is the two-dimensional circle that results from letting α and β rotate freely that needs to be mapped to a one-dimensional string.[1] Given that this information cannot come from any other source than from (i) the lexicon and the (ii) the properties of the syntactic derivation (which are possibly restricted by 'interface readability'), and given that the lexicon is inherently unordered, it follows that the tree somehow must also contain information about order, as expressed by corollary (3).

(3) **COROLLARY**: Trees encode dominance and precedence relations.

(4) **QUESTION:** How can linear order be 'squeezed' from a tree?

● Movement by definition consists in surface reordering, hence a change in precedence relations. The traditional perspective on how to restrict movement (Locality):

---

[1]The dominant node X is not audible, it is the containing category, so these two dimensions need not to be collapsed. The mobile picture as it is used in the literature is misleading, in that it would create a three-dimensional object, which then would have to map to a one-dimensional string.

(5) **LOCALITY AND DOMINANCE**

a. *Locality* is defined in terms of dominance (strong islands) and c-command (weak islands.)

b. c-command is defined in terms of dominance (or containment; see (6)).

c. *Locality* is derived from dominance.

(6) α *c-commands* β iff

(i) α does not dominate β and

(ii) every node dominating α also dominates β.
(alternatively: the mother node(s) of α also dominate(s) β.)

● There are also attempts to derive the order of terminals from a syntactic tree structure fit in this program (Kayne 1994).

(7) **WORD ORDER AND DOMINANCE**

a. *Word order* is defined in terms of asymmetric c-command.

b. c-command is defined in terms of dominance (or containment).

c. *Word order* is derived from dominance.

Again, dominance (containment) is the primary notion.

PREVIEW: In this and the next class, we'll look at ways for deriving word order from trees. Then, we'll turn to recent work (by Williams, Müller, Fox & Pesetsky) that challenges the Dominance Hypothesis, and attempts to derive at least parts of the restrictions on movement from its competitor, the Precedence Hypothesis:

(8) **PRECEDENCE HYPOTHESIS**

Syntactic principles refer to precedence, not c-command/dominance.

But in order to get there, it is still necessary to specify some background assumptions. More concretely the plan looks as follows:

NEXT:

○ Bare Phrase Structure vs. X'-theory

○ The LCA as a mapping operation from 2-dimensional trees to 1-dimensional PF-objects

○ The LCA and Bare Phrase Structure

○ Consequences for the treatment of movement (copies and traces)

---

2. BARE PHRASE STRUCTURE

(9) **GOAL**: Derive basic properties of X'-theory (Chomsky 1994 and later work)

In Bare Phrase Structure (BPS; Chomsky 1995), Merge is the basic structure building operation. Merge combines two objects K and L, forming an ordered pair, and adds the label α:

(10) {α, {K, L}}, where α is the label (Chomsky 1995)

On the assumption that the label is either K or L, Chomsky's notation collapses to the *Wiener-*

Kuratowski method of writing ordered pairs in set notation. Assume moreover that the first member of the pair provides (by convention) the label. Then Merge can be defined as in (11):

(11) **MERGE**(α, β) := {α, {α, β}} <= notational variant of => <α, β>
with α the label of {α, β}

The BPS model has at least four advantages over X'-Theory (the discussion partially follows Lasnik, Nunes & Grohmann 2005: 196ff):

I. Diacritics that denote bar level (° in X°, '' in X', 'P' for XP) are not part of the lexicon ➜ violation of Inclusiveness.

(12) **INCLUSIVENESS**
The computation does not introduce new items apart from the lexical items and their features. (Corollary: all features are part of the lexicon.)

In BPS, the levels are identified by functional determination: just like the notion 'object' and 'subject' are relational notions, that are specified by the syntactic context (SpecTP vs. sister to head), the phrase structural status of a category (maximal, minimal, intermediate) is determined by its respective position w.r.t. other categories. Thus, BPS conforms better with Inclusiveness.

(13) a. α is a maximal projection iff α does not project
b. α is a minimal projection iff α is selected from the numeration
c. α is an intermediate projection iff α is neither a maximal nor a minimal projection.

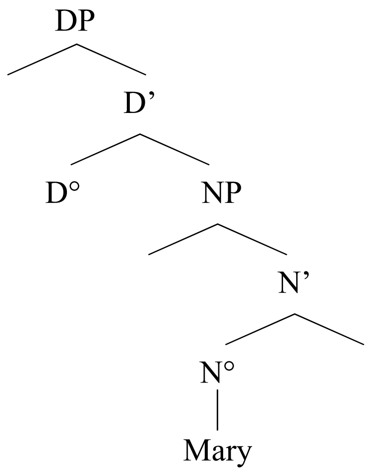
(14) **QUESTION:** What about segments (XP adjunct to XP)? If the lower segment projects, it does not count as maximal, if it does not project, the higher one does not.

➜ Only full categories are visible to syntax. Adjunction structures are complex categories made up of two segments

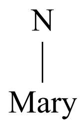
II. BPS derives Endocentricity, which needs to be stipulated in X'-theory by stating that X° projects to X', and not Y' (where ¬X=Y). It does so by the format of Merge in (11).

III. BPS eliminates spurious, superfluous projections by getting rid of branching nodes that only dominate a single contentful node. Instead of being parsed as in (15), a name such as *Mary* is now reduced to (16)a (to be further simplified), while a VP is matched on the tree (16)b:

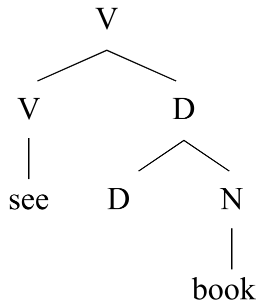
(15)



(16) a.

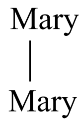


b. see the book



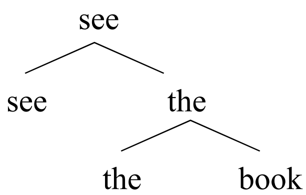
IV. Finally, BPS eliminates categories all together, and thereby reduces a redundancy between lexicon and PS (see discussion of X'-theory, where the change from PS-rules to X'-theory was motivated by a similar search for reducing redundancies.) More specifically, the fact that *Mary* is a noun is already stored in the lexicon. Thus, introducing a label N in the derivation does not add new information. Thus, (16)a is further simplified to (17)a, which can (in fact has to) be collapsed into (17)a' while (16)b is now transposed into (17)b (intermediate representation omitted).

(17) a.



a'. Mary

b.



As an additional bonus, rewriting (17)a as (17)a' eliminates the distinction between *lexical item* and *terminal nodes* - they can now no longer be distinguished.

● Why does (17)a collapse into (17)a'? Merge applies to two terms. The first member of the pair serves a dual function: it denotes a term that merges, and the term that projects (label). The second member of the pair is the non-projecting second term that the first member merges with. Now in trees such as (17)a, *Mary* is the label, so should be the first member of the set. But since it does not merge with anything, the resulting pair is not defined. Merge by definition only applies to two terms.

> NB: *Mary* does not merge with the empty set. This would yield <*Mary*, {}>, or {*Mary*, {*Mary*, {}}}, a branching tree that combines *Mary* and {}, formally an object different from (17)a.

Thus, a system that just admits Merge cannot even represent (17)a. No such problems arise with (17)a', which contains the same syntactic information, but represents a tree theoretic object of its own (Chomsky calls them 'syntactic objects'), and therefore can combine with other syntactic objects.

> NB: Trees are pairs <A, D>, where A is a set of nodes and D a set of pairs capturing the dominance relation. Trees can be trivial in that D is empty: <a, {}>. Thus, *Mary* is a tree/phrase marker.

3. THE LCA

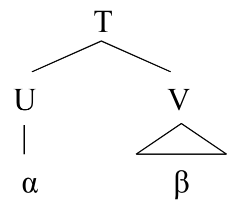
3.1. OBJECTIVES

GOALS A (the main objective): derive basic properties of X'-theory related to ordering generalizations (NB: Kayne 1994 was still written before the advent of BPS; and Kayne was concerned with slightly different properties than Chomsky - see below).

- Specifiers precede heads in all languages that have been studied
- Adjuncts are always adjoined to the left
  - ⇨ requires new analyses for relative clause, postnominal APs, multiple modification, etc....
- Complementation restricted:
  - ○ X° does not take Y° as a complement,
  - ○ XP does not combine with YP

GOAL B (not so central): find an algorithm for mapping structure to order. The fact that the terminal α precedes the terminal β in the tree below does not necessarily mean, that they are *pronounced* in that order! If in the chemical representation C_6H_5OH for Ethanol, the carbon precedes the hydrogen, this does not mean that in the real world object - alcohol molecules - we find such an order. (Recall also the mobile metaphor.)

(18)



3.2. THE IDEA

In a well-formed tree, the terminal nodes must be dominated by non-terminal nodes in a special relation which makes it possible to *unambiguously* identify the ordering of *all* terminals. The relation is **ASYMMETRIC C-COMMAND.** More precisely, there are two conditions:

○ First, for each pair of terminals, it must be possible to find two nodes that dominate these terminals respectively, and which asymmetrically c-commands each other. This can be called the **TOTALITY CLAUSE**, because it requires that *all* pairs satisfy this condition.

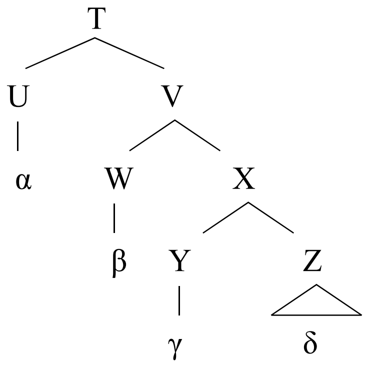
○ Second, there should be no two non-terminals above the terminals that reverse the asymmetrically c-command order. This can be called the **ANTISYMMETRY CLAUSE**.

If a tree meets these requirements, it is licensed by what Kayne (1994) calls the **LINEAR**

CORRESPONDENCE AXIOM (LCA).

● (19) is such a configuration:

(19)



➜ T, ...Z: non-terminal nodes

➜ α,...δ: terminal nodes (lexical items)

○ *Totality Clause* is fulfilled: each pair has asymmetrically c-commanding mothers:

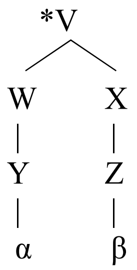
(20) a. α is dominated by U and
b. β is dominated by W and
c. U asymmetrically c-commands W

(21) a. α is dominated by U and
b. γ is dominated by X (or Y) and
c. U asymmetrically c-commands X (or Y)

and so on.....

○ *Antisymmetry Clause* is satisfied: there is no pair of non-terminals dominating, say α and β, respectively, such that in this pair the second member would asymmetrically c-command the first one. To see the workings of this clause more clearly consider structure (22), that violates it:

(22)



➜ non-terminal nodes

➜ terminal nodes (lexical items)

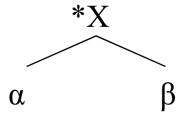
(23) a. α is dominated by W and
b. β is dominated by Z and
c. W asymmetrically c-commands Z

☞ *a mother of α asymmetrically c-commands a mother of β*

(24) a. α is dominated by Y and
b. β is dominated by X and
c. X asymmetrically c-commands Z

☞ *a mother of β asymmetrically c-commands a mother of α*
➜ ✘ Antisymmetry Clause

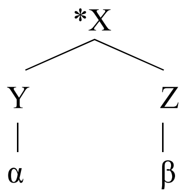
● Finally, trees that violate the Totality clause are usually too small to contain enough mother nodes for the terminals to establish asymmetric c-command:

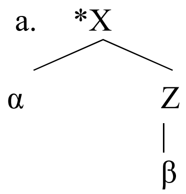
(25)


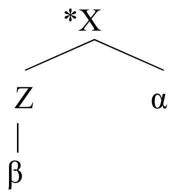
(26) a. α is dominated by X and
b. β is dominated by X but
c. X does not asymmetrically c-commands X
(every node c-commands itself)

➜ ✘ Totality Clause

● Similarly for (27) and (28). Again, in both structures there are no asymmetric c-command relations among the mothers:

(27)


(28) a.


b.


Those are the core cases that the LCA is meant to include. For the moment, they look rather abstract, but once the variables are instantiated by actual labels, the consequences can be easily assessed. Intuitively, all structures have one of two properties: either they are 'too small', i.e. the terminals are not separated from the nodes at which they join the tree by enough structure. This situation leads to violations of the Totality clause. Alternatively, the trees can be too 'symmetric' (in a geometrical sense), triggering Antisymmetry violations.

NEXT:
○ Formal definitions of Kayne
○ Consequences for X'-theory
○ Consequences if BPS is adopted

3.3. FORMAL DEFINITIONS

(29) Domains:
N = set of nodes
N_T = set of terminal nodes
N_{NT} = set of non-terminal nodes

(30) **DOMINANCE AND C-COMMAND**

For any $x,y,z \in N$

a. x DOMINATES y: $x \triangleright^* y$ (antisymmetric, transitive, reflexive)

b. x PROPERLY DOMINATES y: $x \triangleright^+ y =_{Def} x \triangleright^* y \wedge x \neq y$

c. x IMMEDIATELY DOMINATES y: $x \triangleright y =_{Def} x \triangleright^* y \wedge x \neq y \wedge (\forall z)(x \triangleright^* z \wedge z \triangleright^* y \rightarrow z = x)$
(antisymmetric, intransitive, irreflexive)

d. C-COMMAND (C): $xCy =_{Def} \neg x \triangleright^* y \wedge \exists z(z \triangleright x \wedge z \triangleright^+ y)$ (irreflexive)

e. ASYMMETRIC C-COMMAND (A): $xAy =_{Def} xCy \wedge \neg yCx$

(31) **LINEAR CORRESPONDENCE AXIOM (LCA)**
d(A) is a linear ordering of the terminals of the tree

(32) a. The IMAGE d of a non-terminal node x is the set of terminals that x dominates
For any $x \in N_{NT}$, $d(x) =_{Def} \{a \in N_T | x \triangleright^* a\}$

b. The IMAGE d of a pair of nodes:
For any $x,y \in N_{NT}$, $d(<x,y>) =_{Def} \{<a,b> \in N_T x N_T | x \triangleright^* a \wedge y \triangleright^* b\}$

- An slightly more transparent way to formalize the LCA (adapted from Stabler 1997):

(33) **NON-TERMINAL-TO-TERMINAL RELATION T**

For any $a,b \in N_T$,
$aTb =_{Def} (\exists x,y \in N_{NT})(x \triangleright^* a \wedge y \triangleright^* b \wedge xAy)$

(T collects all pairs of terminals which are dominated by non-terminals in asymmetric c-command relation)

(34) **LINEAR CORRESPONDENCE AXIOM (VERSION 2)**
T is a LINEAR ORDER of N_T, i.e. for any $x,y,z \in N_T$ the following holds:

a. $x \neq y \rightarrow xTy \vee yTx$ (TOTALITY/LINEARITY)

b. $xTy \wedge yTz \rightarrow xTz$ (TRANSITIVITY)

c. $xTy \wedge yTx \rightarrow x=y$ (ANTISYMMETRY)

(35) A binary relation R is a LINEAR ORDER iff it satisfies (i) - (iii):

(i) TOTALITY: $(\forall x,y)(x \neq y \rightarrow xRy \vee yRx)$ (= LINEARITY)

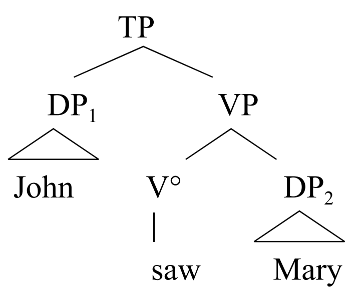
(ii) TRANSITIVITY: $(\forall x,y,z)(xRy \wedge yRz \rightarrow xRz)$

(iii) ANTISYMMETRY: $(\forall x,y)(xRy \wedge yRx \rightarrow x=y)$

3.4. APPLICATION

Consider first the well-formed tree in (36):

(36)



● First, a set is construed - the T-set - which collects pairs of terminals. In particular, these terminals need to fulfill the condition that they are dominated by non-terminals in an asymmetric c-command relation.

(37) <John, saw> $\in T_{TP}$ because
  a. DP_1 dominates *John*
  b. V° dominates *saw* and
  c. DP_1 asymmetrically c-commands V°

(38) <John, Mary> $\in T_{TP}$ because
  a. DP_1 dominates *John*
  b. DP_2 dominates *Mary* and
  c. DP_1 asymmetrically c-commands DP_2

(39) <saw, Mary> $\in T_{TP}$ because
  a. V° dominates *saw*
  b. D_2° dominates *Mary* and
  c. V° asymmetrically c-commands D_2°

..... and nothing else is in the T-set for TP. Thus, the extension of T_{TP} can be given as follows:

(40) T_{TP} = {<John, saw>, <John, Mary>, <saw, Mary>}

● Second, it is verified that the T-set observes the conditions on linear orders. Yes, this is the case: the set in (40) is total because all possible pairings of terminals are members. It is transitive, and it is antisymmetric - the latter because there are no conflicting ordering statements of the form <α,β> and <β,α> in T_{TP}.

➜ Thus, the tree in (36) observes the LCA.

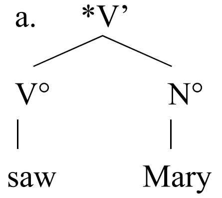
3.5. Consequences for X'-Theory

Next, consider some of the inadmissible trees that are weeded out by the LCA. They establish generalizations about possible X'-structures.

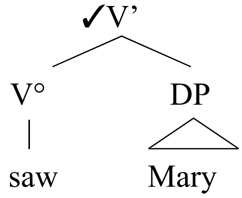
I. Complements of heads

● Complement of a head cannot be a head. More generally, two *heads* can never be sisters.

(41) a.



b.

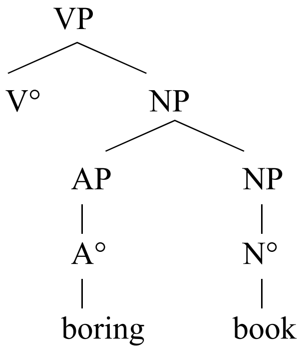


T = {}

➜ ✘LCA due to violation of totality

II. ADJUNCTION

● At first sight, adjunction seems to be inadmissible:

(42)


(43) T_{NP} = {<boring, book>, <book, boring>}

➜ ✘LCA due to violation of antisymmetry

○ Kayne avoids this consequence by changing the definition of c-command:

(44) **ASSUMPTION:** Segments do not count for the computation of c-command.

(45) α *c-commands* β iff

(i) α and β are categories (not just segments thereof)
(ii) no segment of α contains β (= α excludes β)
(iii) every category dominating α also dominates β.

As a consequence, the tree in (42) is admissible:

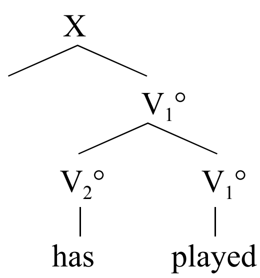
(46) <boring, book> ∈ T_{TP} because
a. AP dominates *boring*
b. NP dominates *book* and
c. AP asymmetrically c-commands NP

(47) <book, boring> ∉ T_{TP} because
a. AP dominates *boring*
b. NP dominates *book* and
c. NP contains AP, and therefore does not c-command AP
d. N° dominates *book* and
e. N° does not c-command AP

(48) T_{NP} = {<boring, book>}

➜ ✓LCA

● Head adjunction is also licensed by the revised version of c-command.

(49)


(50) a. V_1 does not c-command V_2 because V_1 does not exclude V_2.
b. V_2 however c-commands V_2 because X is the first category that dominates V_2 , and X also dominates V_1.

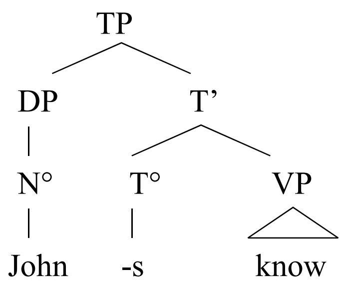
(51) $T_{V1^{\circ}}$ = {<has, played>}
➜ ✓LCA

The reason why (42) failed to conform with the LCA was that two non-heads are construed as sisters. More generally, such a configuration is always blocked by the LCA. The LCA restricts sisterhood to structures in which one sister is a head, and the other a non-head. This has interesting consequences for the way in which specifiers are parsed.

III. Specifiers

Consider a tree that contains an intermediate projection T’: it inadvertently violate the LCA.

(52)



(53) <John, -s> $\in T_{TP}$ because
a. DP dominates *John*
b. T° dominates *-s*
c. DP asymmetrically c-commands T°

(54) <-s, John> $\in T_{TP}$ because
a. N° dominates *John*
b. T’ dominates *-s*
c. N° asymmetrically c-commands T’

(55) T_{TP} = {<John, -s>, <-s, John>, ...}
➜ ✘LCA due to violation of antisymmetry

(56) **Corollaries:** ○ There are no intermediate (X’) levels.
○ Specifiers are adjuncts.

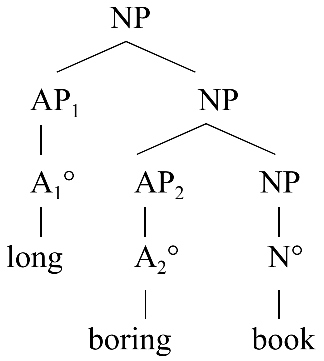
○ An alternative (roughly Chomsky 1999): LCA only sees X° and XP,but not X’

IV. Number of adjuncts

The LCA has a final important consequences: it prohibits multiple adjunction of XPs as well as of heads. Interestingly, the reasons why these two types of trees are blocked are not the same.

● Consider multiple XP-adjunction first:

(57)



(58) <long, boring> ∈ T_{TP} because
a. AP_1 dominates *long*
b. AP_2 dominates *boring* and
c. AP_1 asymmetrically c-commands AP_2 (segments don't count!)

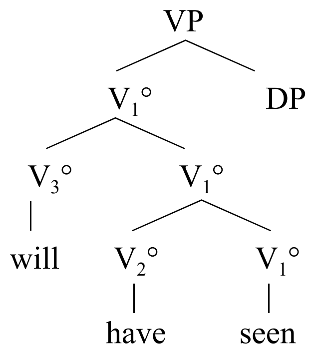
(59) < boring, long> ∈ T_{TP} because
a. AP_1 dominates *long*
b. AP_2 dominates *boring* and
c. AP_2 asymmetrically c-commands AP_1 (segments don't count!)

(60) T_{NP} = {<boring, book>, <book, boring>}
➜ ✘LCA due to violation of antisymmetry

(61) **ASSUMPTION:** AP_1 adjoins to AP_2 (rather dubious)

● Next, multiple X°-adjunction is blocked by the *totality* - not by the antisymmetry - requirement:

(62) will have seen....



V_1 ° does not c-command any of the other heads, because it does not exclude them. But V_3 and V_2 are in a symmetric c-command relation: for both, the first dominating category is VP, which in turn also dominates the other head. It follows that the two heads *will* and *have* cannot be ordered.

(63) T_{V1} = {<have, seen>}
➜ ✘LCA due to violation of totality (*will* and *have* are not ordered)

3.6. Reducing dimensions (GoalB)

Recall that the definition of c-command is inherently blind to precedence. The fact that in a tree, the T relation e.g. is specified as {<a,b>, <b,c>, <a,c>} does not say anything about whether the T-relation translates as precedence or subsequence or some other, more complex linearization relation. Imagine e.g. a backwards-talking competition among kids, where *The book sucks* has to be rendered as *sucks book the*. In such a competition, the speakers would presumably still create the strings in the normal order first, and only map them to the backwards order afterwards.

The potential point of confusion here is that *linear order* is a technical (mathematical) term, which describes properties of certain relations - it does not entail that sets of pairs which are *linear orders* in this technical sense are also linearized in the same way by the PF interface.

➜ Kayne suggests that left to right temporal sequencing is an inherent property of language (not the strongest motivation). Assuming that the mapping from the linear order established by the T-set to order in the tree is indeed defined in terms of precedence (i.e. the T-relation is be mapped onto precedence relation) the following consequences for legitimate phrase structure trees ensue:

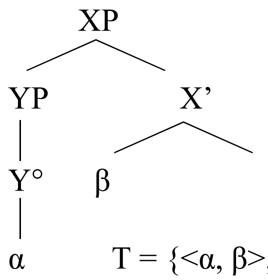
Consequences: hierarchically higher categories precede lower ones

- ○ specifiers precede heads
- ○ heads precede adjuntcs
- ○ modifiers precede heads

Epilogue: Uriagereka (1999)

Proposes alternative way for deriving order relation in terms of **Multiple Spell Out**.
Recall again the reason why X'-levels must not be part of the representations in the LCA: X' could asymmetrically c-command into YP (Y°), creating a ordering conflict:

(64)



T = {<α, β>, <β, α>,...}

➜ ✘LCA due to violation of antisymmetry

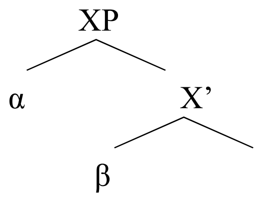
Uriagereka suggest to keep intermediate projections and offers an interesting way to derive linearization. The account relies in two ingredients:

○ Specifiers are spelled out separately and are treated like terminals. (Otherwise, they could not be ordered.) This collapses YP into α.

○ C-command only defined for terminals.

As a consequence, the new representation observes the LCA, and at the same time derives the order α>β. (For more details see e.g. Lasnik & Uriagereka 2005: 48ff)

(65)

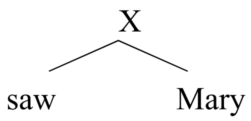


4. Some complications

4.1. Bare phrase structure

Combining the LCA with or integrating into BSP leads to incompatibilities. The BPS representation below violates the LCA, as the two heads cannot be ordered (totality):

(66)

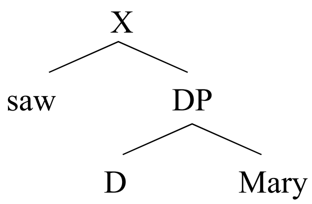


(where X is the label - possibly *saw*)

Possible solution (from Hornstein et. al 2005: 229)

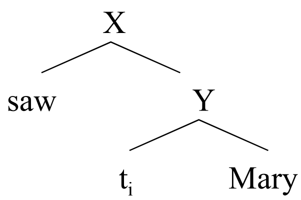
- Add structure and assume that LCA only sees *overt* categories:

(67)



- Movement of one of the lexical items results in asymmetric c-command, and totality is observed:

(68)



Question: But what about the status of the trace? Would it not induce LCA violation?
Answer: Not if only *overt* categories are visible to LCA.

4.2. Movement, traces and copies

Problem I

The assumption that *overt* categories are visible to LCA is subject to debate, though. In particular, it does not align well with the Copy Theory of movement. More precisely, at the moment at which the LCA applies, the copies/traces are not distinguished from their antecedents: they are just occurrences in a chain. Ideally, the fact that they end up as silent should be derived, and not assumed as a primitive. Thus, the solution to the problem in (68) is only suboptimal.

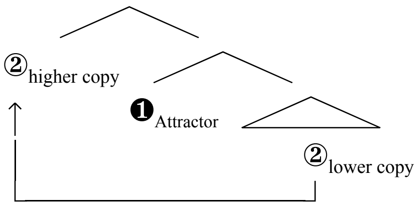
An alternative solution presents itself, though: copies are internally complex, hence contain structure. This property (together with the remarks in 4.1.) help to construe the relation between the head and the copy as asymmetric.

PROBLEM II

But not only the relation between the copy/trace and the head proves problematic. The relation among the copies equally triggers LCA violations - this time in offense of the symmetry clause.

Suppose e.g. that category ② moves across an attracting head. Then, prior to movement, the attractor ❶ in (69) is dominated by a non-terminal which asymmetrically c-commands the lower copy of the attractee ② prior to movement; subsequent to raising of ② to the left of ❶, the higher occurrence of ② is dominated by a non-terminal asymmetrically c-commanding ❶. Since the T-set now contains <②,❶> as well as <❶,②>, the terminals embedded inside the movement copies cannot be ordered, inducing an LCA violation.

(69) T = {<②,❶>, <❶,②>, ...} ✘LCA



NB: movement is allowed by LCA to leave copies if it proceeds string vacuously. This is so at least if the order relation (linear order) is a so called *weak order*, which is transitive, total, reflexive and antisymmetric. (But there are also strong orders, which are transitive, irreflexive and asymmetric).

Analyses that address problem II lead to interesting new insights into how the LCA can be employed in the derivation of properties of movement:

4.3. CONSEQUENCES OF LCA FOR MOVEMENT

I. DETERMINING PF POSITION OF COPY

Nunes (1995, 1999) uses this generalization to derive the fact that only one copy is pronounced. He assumes that LCA is a condition on linearization of *overt categories*. Thus, foregoing Spell-Out (or eliding) a copy - ②$_{\text{lower copy}}$ in (69) - resolves the conflict.

The analysis is further developed into account why higher copy must delete: features that drive movement are checked in the higher copies, which therefore have fewer features. Deletion of fewer features is more economical than deletion of more ⇨ higher copy deletes.

II. LOCALITY AND PHASES

The necessity to re-align structures like (69) with the LCA can seen as a sign that the LCA only sees parts of the derivations, and therefore must be relativized to units such as phases. Such approaches can, among others, account for Locality Superiority effects (Lechner 2001, 2004). Assume that each phase has its own T-set. If movement leaves a phase, as in (70)a, it may cross other LCA visible material. If it needs to pass at the edge of the phase, though, as in (70)b, the initial short movement induces an LCA-violation inside the lower phase.

(70) a. LICIT MOVEMENT

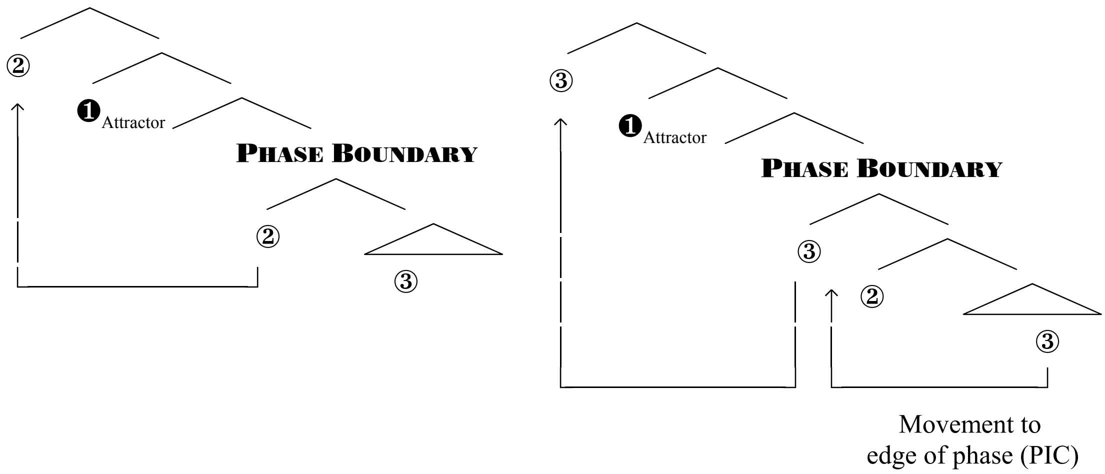
$T_{\text{lower Phase}}$ = {<②,③>} ✔LCA

$T_{\text{higherPhase}}$ = {<②,❶>} ✔LCA

b. MLC VIOLATION

$T_{\text{lower Phase}}$ = {<②,③>, <③,②>} ✘LCA

$T_{\text{higher Phase}}$ = {<③,❶>} ✔LCA



This setup ensures that a phase-external head can only attract the highest suitable category in the subordinate phase (② in (70)). But this prohibition constitutes nothing else but the core of the **MINIMAL LINK CONDITION**, which demands attraction of the closest feature-compatible candidate.

---

5. SUMMARY: REPAIR STRATEGIES FOR LCA

An LCA violation can be amended by:

- Addition of (functional, empty) structure
- Movement
- PF-deletion of all but one copy (Nunes 1999)
- Relativization of the LCA to subdomains (Richards 2001; Lecher 2004)
- Multiple Spell-Out (Uriagereka 1999)

(71) **OBSERVATION**: Crucially for future purposes, it is possible to express *locality conditions on movement* by *linearization conditions on copies*.

BIBLIOGRAPHY

Ernst, Thomas. 2001. The syntax of adjuncts. Cambridge: Cambridge University Press.

Kayne, Richard S. 1994. The Antisymmetry of Syntax. Cambridge, Massachusetts: MIT Press.

Lasnik, Howard and Juan Uriagereka. 2005. A course in minimalist syntax. Malden: Blackwell.

Lechner, Winfried. 2004. Extending and Reducing the MLC. In A. Stepanov, G. Fanselow and R. Vogel (eds.), Minimality Effects in Syntax, pp. 205-241. Berlin, New York: Mouton de Gruyter.

Hornstein, Norbert, Jairo Nunes and Kleanthes Grohmann. 2005. Understanding Minimalism. Cambridge: Cambridge University Press.

Nunes, Jairo. 1999. Linearization of chains and phonetic realization of chain links. In S. Epstein and N. Hornstein (eds.), Working minimalism, pp. 217-49. Cambride, Mass: MIT Press.

Richards, Norvin. 2001. A Distinctness Condition on Linearization.

Stabler, Edward. 1997. Derivational Minimalism. Ms., Los Angeles: UCLA.

Uriagereka, Juan. 1999. Multiple Spell-Out. In S. Epstein and N. Hornstein (eds.), Working minimalism, pp. 251-280. Cambride, Mass: MIT Press.