Rethinking Linearization[1]

Kyle Johnson
University of Massachusetts at Amherst
February 2017

In a series of papers, a book and a dissertation, Jairo Nunes has provided a compelling way of deriving a signature property of movement, a property I will call "terseness."

(1) Terseness
When a term is moved from one position to another, it gets spoken in only one of those positions.

There are exceptions to Terseness, and some of these Nunes' account predicts. This venue doesn't provide the space to consider these exceptions, or how they fit Nunes' project, so I will set them aside and concentrate on the normal case, in which Terseness holds. Nunes' leading idea is that movement creates a structure that the linearization algorithm can interpret only if Terseness holds.

Nunes' account has two parts. First, he adopts the Copy Theory of Movement (2).

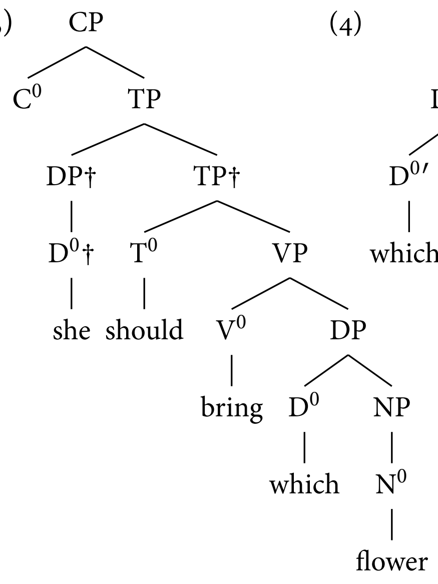
(2) Copy Theory of Movement
  a. From a term X is made a copy: X′
  b. X′ is merged into a position higher than X

On this view, Movement could take the structure in (3), form a copy of *which flower* and form the structure in (4).[2]

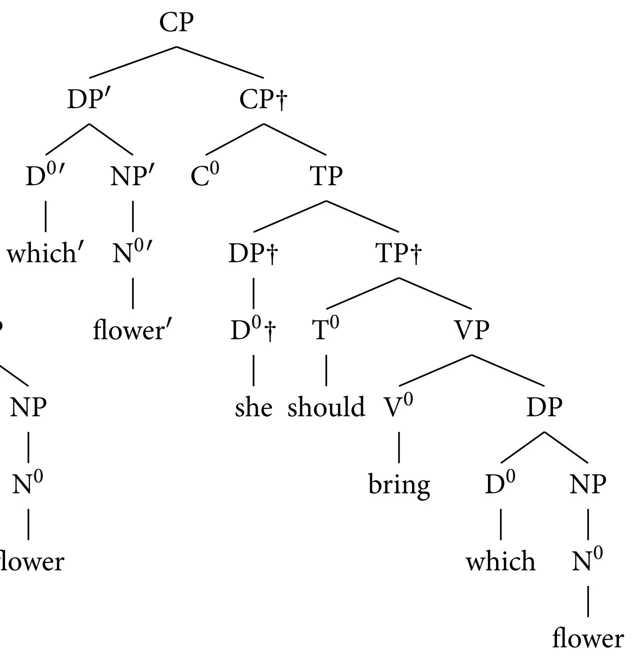
1 This paper is largely a channeling of many people's thoughts. These include Sakshi Bhatia, David Erschler, Hsin-Lun Huang, Rodica Ivan, Jyoti Iyer, Petr Kusily, Deniz Ozyildiz, Ethan Poole, Katia Vostrikova, Michael Wilson, Rong Yin, Rajesh Bhatt, Peggy Speas, Nikolaos Angelopoulos, John Gluckman, Nicoletta Loccioni, Travis Major, Iara Mantenuto, Sozen Ozkan, Richard Stockwell, Carson Schutze and Tim Hunter. A special thanks to Leland Kusmer whose guidance has improved this paper in many ways.

2 In order to focus on just one movement operation at a time, I will only consider cases of embedded constituent questions, where movement of the T^0 to C^0 doesn't occur.

(3)



(4)



The second part relies on a standard condition on how phrase markers are linearized into strings that Kayne (1994) calls Antisymmetry.

(4) Antisymmetry
A linearization cannot contain both $a < b$ and $b < a$.

Antisymmetry assumes that a linearization is a set of ordered pairs $x < y$, where x and y are words and "<" is the precedence relation. Antisymmetry simply states that no word can both follow and precede another. Nunes' second proposal, then, is that Antisymmetry cannot distinguish one word from its copy. The structure in (4) is not pronounced with two instances of *which* and *flower* because a linearization that contains both *which′<she* and *she<which* will be a violation of Antisymmetry. This is Terseness.

One goal of this paper is to define copies so that they have the effect of invoking Antisymmetry in the way that Nunes envisions. That definition will use the idea broached in Engdahl (1980) that a moved term is a term in two syntactic positions.[3] This can be represented by letting phrase marker trees allow multidominance. Another goal of this paper is to devise a linearization algorithm that can handle such trees.

3 Engdahl cites the unpublished Peters and Ritchie (1981) as her source for the idea

1 Nunes Proposal

Nunes couches his idea with a slightly modified version of the linearization algorithm in Kayne (1994). The key departure from Kayne's algorithm concerns the items that are linearized. Kayne's algorithm linearizes morphemes – including subword material – and Nunes' doesn't. I'll adopt Nunes' view, which is useful in accounting for certain exceptions to Terseness. A goal of Kayne's work is to derive (5) from the linearization algorithm.

(5) If XP asymmetrically c-commands YP, then the words dominated by XP (=d(XP)) will precede the words dominated by YP (=d(YP)) (modulo the effects of movement).

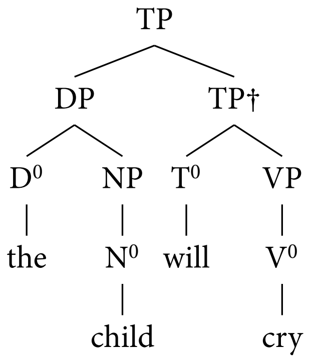
(6) α c-commands β if every phrase dominating α dominates β too, and α doesn't dominate β. α asymmetrically c-commands β if α c-commands β and β doesn't c-command α.

This is achieved by building (5) into the linearization algorithm along the lines of (7).

(7) a. Let L be the set of pairs of heads and phrases, $<A, B>$, in a phrase marker P such that A asymmetrically c-commands B.

b. The linearization of P is the union of d(A) < d(B) for every $<A, B>$ in L.[4]

As Kayne notes, (7) needs to be weakened if it is to work for phrase markers that have Specifiers. To see this, consider how (7) applies to (8).

(8)


The L for (8) is (9a), and this produces the linearization in (9b).

4 More explicitly: the linearization of P is $\{a < b: \forall a \in d(A) \text{ and } \forall b \in d(B) \text{ if } <A, B> \text{ is in } L \text{ of P}\}$. Note that "<" is the precedes relation.

(9) a. L = <D^0,N^0>, <DP,T^0>, <DP,VP>, <DP,V^0>, <T^0,V^0>, <TP†,D^0>, <TP†,N^0>

b. $$\left\{\begin{array}{lll} the < child & child < can & can < cry \\ the < can & child < cry & \\ the < cry & & can < the \\ & & cry < the \\ & & can < child \\ & & cry < child \end{array}\right\}$$

≡ can cry the child can cry

(9b) violates what Kayne calls Antisymmetry.

(10) Antisymmetry
A linearization cannot contain both *a*<*b* and *b*<*a*.

The problem with (7) is that it allows too many asymmetric c-commanding pairs to enter *L*. Because TP† is part of some of the pairs in *L*, the orderings *can*<*the*, *can*<*child*, *cry*<*the* and *cry*<*child* get into the linearization. But because DP is also part of some of the pairs in *L*, the linearization contains *the*<*can*, *the*<*cry*, *child*<*can* and *child*<*cry*. To address this problem, Kayne proposes a way of limiting the class of items that can be in *L* so that it achieves certain goals his system has for ordering sub-word morphemes. Because that is not a feature of the procedure needed to derive Terseness, I will take a slightly different tack. I will limit *L* to just maximal and minimal projections.

(11) a. Let *L* be the set of pairs of heads and maximal projections, <*A*, *B*>, in a phrase marker P such that *A* asymmetrically c-commands *B*.
b. The linearization of P is the union of d(*A*) < d(*B*) for every ordered pair in *L*.

Because TP† is neither a minimal nor a maximal projection it will be jettisoned from *L*. (11) will produce the *L* in (12a), and this generates the correct linearization in (12b).

(12) a. L = <D^0,N^0>, <DP,T^0> <DP,VP>, <DP,V^0>, <T^0,V^0>

b. $$\left\{\begin{array}{lll} the < child & child < can & can < cry \\ the < can & child < cry & \\ the < cry & & \end{array}\right\}$$

≡ the child can cry

(11) correctly linearizes a wide array of syntactic structures and provides a way of deriving (5).

We are now ready to see how Nunes proposes to derive Terseness. His proposal amounts to adopting (13).

(13) A term, X, and its copy, X′, cannot be distinguished by Antisymmetry.

A consequence of (13) is that a linearization which contains both X<Y and Y<X′ will violate Antisymmetry. Applying (11) to the result of movement in (4) produces the linearization in (14b).

(14) a. *L* = <D$^{0\prime}$,N$^{0\prime}$>, <DP′,C^0>, <DP′,TP>, <DP′,DP†>, <DP′,D^0†>, <DP′,T^0>, <DP′,VP>, <DP′,V^0>, <DP′,DP>, <DP′,D^0>, <DP′,NP>, <DP′,N^0>, <C^0,DP†>, <C^0,D^0†>, <C^0,T^0>, <C^0,VP>, <C^0,V^0>, <C^0,DP>, <C^0,D^0>, <C^0,NP>, <C^0,N^0>, <DP†,T^0>, <DP†,VP>, <DP†,V^0>, <DP†,DP>, <DP†,D^0>, <DP†,NP>, <DP†,N^0>, <T^0,V^0>, <T^0,DP>, <T^0,D^0>, <T^0,NP>, <T^0,N^0>, <V^0,D^0>, <V^0,NP>, <V^0,N^0>, <D^0,N^0>

b.

$$\left\{\begin{array}{llllll}
which' < flower' & flower' < should & should < she & she < bring & bring < which & which < flower \\
which' < should' & flower' < she & should < bring & she < which & bring < flower & \\
which' < she & flower' < bring & should < which & she < flower & & \\
which' < bring & flower' < which & should < flower & & & \\
which' < which & flower' < flower & & & & \\
which' < flower & & & & &
\end{array}\right\}$$

≡ which′ flower′ should she bring which flower

Because of the existence of *which*'<*bring* and *bring*<*which* in (14b), along with many other such pairs, Antisymmetry is violated.

This derives the impossibility of speaking a moved term in both of the places it occupies, but something more is needed to produce the string that actually arises. Nunes suggests that this involves a movement-specific deletion operation which removes orderings from a linearization. Applied to (14), this deletion operation could remove orderings to form one of the strings in (15), all of which satisfy Antisymmetry.

(15) a. which flower should she bring
b. which should she bring flower
c. flower should she bring which
d. should she bring which flower

Nunes assumes, and so shall I, that (15a) and (15d) are possible outcomes – some languages choosing one or the other – but that (15b) and (15c) are not. To block these two outcomes, Nunes makes two assumptions. First the deletion operation in question applies not to a linearization – it doesn't remove elements of the set in (14) for instance – but to the syntactic structure being linearized. It removes the linearization statements corresponding to the phrases and heads that populate a syntactic representation. I'll formulate Nunes' condition, which he calls Chain Reduction, to reflect this.

(16) Chain Reduction
Chain Reduction applied to d(X) deletes every ordered pair in a linearization that contains a word in d(X), X a head or phrase.

To form the strings in (15), Chain Reduction will delete from *L* the ordered pairs indicated in (17).

(17) a. To form (15a), Chain Reduction applies to d(DP).
b. To form (15b), Chain Reduction applies to d(NP′) and d(D^0).
c. To form (15c), Chain Reduction applies to d(D^{0}’) and d(NP).
d. To form (15a), Chain Reduction applies to d(DP’).

The second assumption Nunes makes is that there is an economy condition that favors fewer targets for Chain Reduction.

(18) Economy
Let *N* be the number of terms that an instance of Chain Reduction, *R*, applies to. Block *R* if its *N* is greater than the *N* for another *R* that satisfies Antisymmetry.

Economy will block the applications of Chain Reduction in (17b) and (17c) because of the equally Antisymmetry compliant applications of Chain reduction in (17a) and (17d).

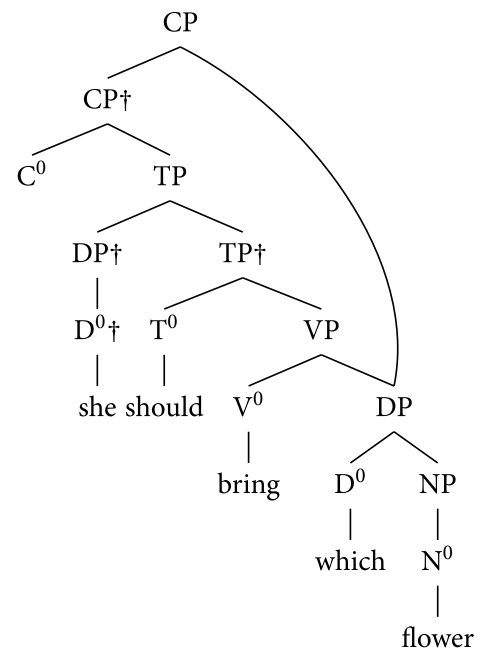
There are a variety of successes for this method of deriving Terseness, and I will not challenge it. Instead, I will focus on understanding (13). Why is Antisymmetry unable to distinguish a term from its copy?

2 Multidominance

A simple way of explaining why a term and its copy are the same thing for Antisymmetry is that they are the same thing. Rather than modeling movement as an operation that creates a copy of a term and puts that term in an additional position, we could model movement as an operation that puts one term in two positions. This is a thesis that Engdahl (1980), Starke (2001), de Vries (2007), Gärtner (2002), among others, have suggested.

An immediate problem with this view, though, is that it leads to the expectation that the denotation a phrase has will be the same in both of the positions movement relates it to. Consider, for instance, a way of representing this thesis that allows one term to have two positions in a phrase marker. That would give (4) the representation in (19).

(19)

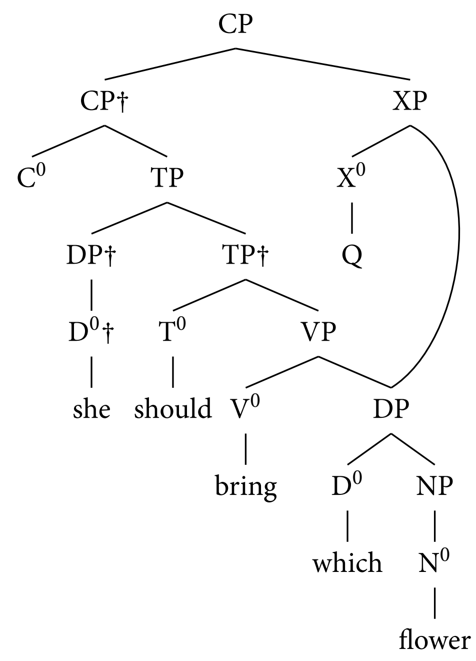


There is evidence that the semantics of constituent questions of this kind must be able to involve a binder/variable relation. In principle, we want phrasal movement to be able to cause a moved phrase to bind a variable in the position it moves from. The representation in (19) makes that possibility obscure. The single phrase, *which flower*, would not seem to be able to simultaneously have the meaning of a variable and the meaning of the term that binds that variable.[5] We want to define "copy of" so that it gives the equivalent of (19) for Antisymmetry, but not for the meanings involved.

In Johnson (2012), I argue that the solution to this dilemma comes from recognizing that there can be material in the higher position that is not part of the term that has moved. If we represent this additional material with "Q," then (19) can be replaced by (20)

5 But see Engdahl (1986) for a method.

(20)



Depending on the kind of semantic relation involved, we can credit the denotation of Q^0 with being responsible for creating a binder out of the higher phrase. See Johnson (2012) for details. I will assume that movement is an operation that puts one term in two positions, but that it does so always in a way parallel to (20). The moved item is part of a larger term in the higher position.

Adopting this view requires a recasting of Nunes' method of deriving Terseness. We cannot rely on an operation like Chain Reduction to fix the violations of Antisymmetry that movement will create as it will overshoot. To see this, consider how (11) will apply to (20); it produces the linearization in (21).

(21) a. L = <X^0,D^0>, <X^0,NP>, <X^0,N^0>, <XP,C^0>, <XP,C^0>, <XP,TP>, <XP,DP†>, <XP,D^0†>, <XP,T^0>, <XP,VP>, <XP,V^0>, <C^0,DP†>, <C^0,D^0†>, <C^0,VP>, <C^0,V^0>, <C^0,DP>, <C^0,D^0>, <C^0,NP>, <C^0,N^0>, <DP†,T^0>, <DP†,VP>, <DP†,V^0>, <DP†,DP>, <DP†,D^0>, <DP†,NP>, <DP†,N^0>, <T^0,V^0>, <T^0,DP>, <T^0,D^0>, <T^0,NP>, <T^0,N^0>, <V^0,D^0>, <V^0,NP>, <V^0,N^0>, <D^0, N^0>

b.

$$\left\{\begin{array}{llllll}
Q < which & which < flower & flower < she & she < should & should < bring & bring < which \\
Q < flower & which < she & flower < should & she < bring & should < which & bring < flower \\
Q < she & which < should & flower < bring & she < which & should < flower & \\
Q < should & which < bring & flower < which & she < flower & & \\
Q < bring & which < which & flower < flower & & &
\end{array}\right\}$$

≡ Q which flower should she bring which flower

There are numerous violations of Antisymmetry in (21b) (e.g., *which<bring* and *bring<which*) as well as the arguably anomalous *which<which* and *flower<flower*. For Chain Reduction to remove these violations, it would have to apply to either d(XP) or d(DP). If it applies to d(DP), (21) will lose all ordered pairs that have either *which* or *flower* in them, producing a linearization that is equivalent to (22).

(22) Q she should bring

If movement puts one thing in two places, thereby explaining (13), then something must replace Chain Reduction in Nunes' explanation for Terseness.

A minimal modification of Nunes' system would be to allow the pairs that go into *L* to be partial in a way that mimics Chain Reduction. Rather than removing ordering statements that produce a violation of Antisymmetry, we can allow the linearization to avoid introducing them to begin with. (11) becomes (23).

(23) a. Let *L* be a set of pairs of heads and maximal projections, <*A*, *B*>, in a phrase marker P such that *A* asymmetrically c-commands *B*.
b. The linearization of P is the union of d(*A*) < d(*B*) for every ordered pair in *L*.

Unlike (11a), which required that *L* contain <*A*, *B*> for every *A* that asymmetrically c-commands *B*, (23a) allows *L* to contain a proper subset of such ordered pairs: all it requires is that *L* contain <*A*, *B*> only if *A* asymmetrically c-commands *B*. (23) allows partial orderings, and so it will have to be coupled with something that ensures that every word in a syntactic representation end up in the linearization. This can be achieved by adopting another of Kayne (1994)'s well-formedness conditions:

(24) Totality
If *a* and *b* are words in P, then either $a < b$ or $b < a$ must be in the linearization of P.

(23) will allow for the English linearization of (20) – in (25) – and Totality will prevent incomplete outcomes like (22).

(25) a. L = <X^0, D^0>, <X^0, N^0>, <D^0,N^0>, <XP,C^0>, <XP, DP†>, <XP,T^0>, <XP,V^0>, <C^0,DP†>, <C^0,T^0>, <C^0, V^0>, <DP†,T^0>, <DP†,V^0>, <T^0,V^0>
b.

$$\left\{\begin{array}{lllll} Q < which & which < flower & flower < she & she < should & should < bring \\ Q < flower & which < she & flower < should & she < bring & \\ Q < she & which < should & flower < bring & & \\ Q < should & which < bring & & & \\ Q < bring & & & & \end{array}\right\}$$

≡ Q which flower she should bring

Moreover, (23) will also correctly block (15b) and (15c), in which *which* and *flower* are linearized in non-contiguous positions. This is because for Totality to be satisfied, XP must be in L. Only if XP is in L will Q get linearized with all the words that are not in XP. But once XP is in L, all of the words in XP (i.e., *Q*, *which* and *flower*) will be linearized in the same way to every word not in XP. A feature of (23) is that it enforces contiguity on any phrase that enters L.[6]

(26) Contiguity
A linearization is contiguous if for every phrase, XP, in L, if $b \notin$ d(XP), then $b < a$ or $a < b$ for every $a \in$ d(XP).

An interesting feature of movement is that it creates structures which violate a stronger form of Contiguity, one that holds of every phrase in a structure, not just those used to form a linearization. This stronger form of Contiguity is quite widely honored by linearization; we should have an account for why it is relaxed just for movement structures. (23) takes a step towards doing this by letting Contiguity hold not of the entire phrase marker, but of the subset of phrases chosen from that phrase marker to base a linearization on. Totality forces this subset to be sufficiently representative, spreading Contiguity among the non-moved parts of the phrase marker. The moved parts of a phrase marker are allowed to violate Contiguity because there is a way of satisfying Totality without considering all the positions they are in.

Unfortunately, this feature of (23) prevents any other linearization of (20), including the one Nunes' theory countenanced in (27).

(27) Q she should bring which flower

In general, if phrasal movement creates a structure in which, like (20), the moved phrase is part of a larger phrase in the higher position, then (23) will not allow covert movement.

What this section shows is that it's possible to preserve much of the linearization algorithm that Nunes uses to explain Terseness, while giving a natural and simple explanation for why Antisymmetry should treat a moved term as if it's one thing in two positions. Kayne called his linearization algorithm the "linear correspondence axiom," or LCA. Let's know this modified version of his algorithm as the "multidominant-friendly linear correspondence axiom," or MLCA.

6 There is a very close resemblance between Contiguity and the central condition in Lisa Selkirk's Match Theory (Selkirk 2011), which requires that phrases map onto prosodic units that contain every word within them. A tantalizing prospect is to reduce Contiguity to this condition on the syntax/prosody mapping.

(28) MLCA

a. Let L of P consist of pairs of minimal and maximal projections, $<A, B>$, where A asymmetrically c-commands B in P.

b. A linearization of P is the union of d(A)<d(B) for every $<A, B>$ in L of P.

c. $d(\alpha) =_{def.}$ all the words dominated by α.

(29) Antisymmetry
A linearization of P cannot contain both $a < b$ and $b < a$.

(30) Totality
A linearization of P must contain $a < b$ or $b < a$ for every pair of words a, b in P.

The mcla has properties which should be regarded as features. Some of them are (31).

(31) mlca Features

a. Preserves the goal of Kayne's lca, i.e. the generalization in (5).

b. Enforces Contiguity on a moved phrase (i.e., blocks (15b) and (15c)).

c. Derives Terseness.

d. Produces linearizations corresponding to overt movement.

It also has a property that could be regarded a bug. If movement has the properties I argued for in Johnson (2012), then it will not allow for a linearization that corresponds to covert movement. I regard that as a bug, and so I will offer an alternative linearization scheme in the next section.

3 Paths

If a structure like (20) is to be able to linearize into covert movement, i.e. a string in which *which flower* follows *bring*, then it will be necessary to allow *Q* and *which flower* to end up non-contiguous. This means that the linearization algorithm cannot prevent *Q* from getting into the linearization unless everything else in d(XP) gets ordered the same way to the things that XP asymmetrically c-commands. We must let *Q* get into the linearization without using XP's position to do so. I cannot see a way of doing that which preserves Kayne's program, so I will abandon (5) as a goal of the linearization scheme.[7] What shouldn't be abandoned, though, is Contiguity which seems to be a general truth about how syntactic structures map onto strings. If movement employs multidominant representations, Contiguity must be relaxed, but only just where multidominance arises. So my goal will be to devise a linearization algorithm which preserves Contiguity in all those cases where multidominance

7 See Abels and Neeleman (2012) for another direction to pursue.

(aka movement) doesn't arise and explain why it selectively permits violations where multidominance does arise.

Contiguity is typically conceived of as a relationship between dominance relations and contiguous strings and this is how I've stated it in (26). It enforces the law in (32).

(32) If words $a_1,\ldots,a_n$ are dominated by a phrase XP (=d(XP)), then $a_1,\ldots,a_n$ will form a contiguous substring in the linearization.

For standard phrase markers that don't have multidominance in them, an equally valid way of stating the law that Contiguity enforces is (33).

(33) If phrase XP_1 dominates phrase XP_2, then the words in XP_2 (i.e., $d(XP_2)$) will form a contiguous substring of the string formed by the words in XP_1 (i.e., $d(XP_1)$).

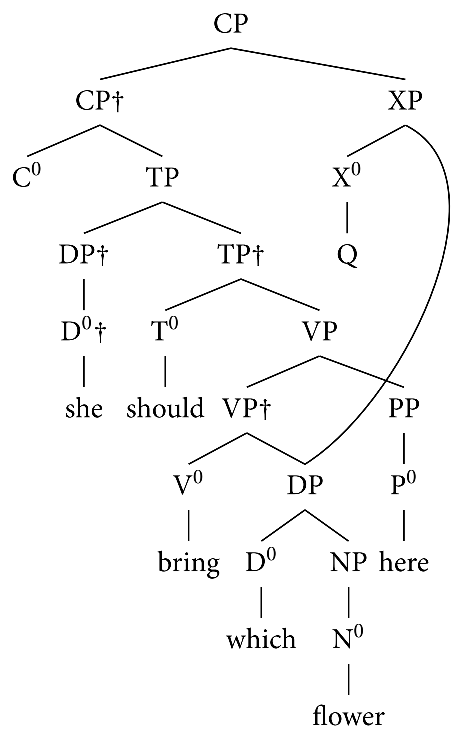
Indeed, the transitive closure of (33) holds for phrase markers that obey Contiguity and don't contain multidominance.

(34) Let p=(XP_1, XP_2,…,XP_n) be a series of phrases such that every XP_i in p is dominated by every $XP_{j\leq i}$ in p. For every p in a phrase marker, $d(XP_i)$ must be a contiguous substring of $d(XP_{j\leq i})$ for every XP in p.

(NB: "dominance" and "substring" are reflexive.)

I will call a series of phrases that form a p, a "path."

Interestingly, (34) isn't obeyed in a phrase-marker that allows for multidominant representations. To see this, consider (35) and the linearization of (35) that corresponds to overt movement, in (36).

(35)


(36) *overt movement linearization:*
Q which flower she should bring here

Two paths that contain DP and NP in (35) are (37).

(37) a. *paths for NP:*
i. (NP,DP,VP†,VP,TP†,TP,CP†,CP)
ii. (NP,DP,XP,CP)

b. *paths for DP:*
i. (DP,VP†,VP,TP†,TP,CP†,CP)
ii. (DP,XP,CP)

(36) makes (37a-i) and (37b-i) violate (34); neither *flower* (=d(NP)) nor *which flower* (=d(DP)) are contiguous substrings of d(TP) (=*she should bring which flower here*), d(TP†) (=*should bring which flower here*), d(VP) (=*bring which flower here*) or d(VP†) (=*bring which flower*). If Contiguity were to be expressed in a way that derives (34), then only covert movement operations would be permitted. That's not a desirable outcome. Notice, however, that if the paths in (37a-i) and (37b-i) are ignored, the linearization in (36) doesn't violate (34). Conversely, the paths in (37a-ii) and (37b-ii) violate (34) if the linearization is (38).

(38) Q she should bring which flower here

Under this linearization, neither d(NP) (=*flower*) nor d(DP) (=*which flower*) are contiguous substrings of d(XP) (=*Q which flower*). This linearization doesn't violate (34), however, if the paths in (37a-ii) and (37b-ii) are ignored. Paths give us a way, then, of linearizing a phrase that is in two positions in either one of those positions. We can use paths to make movement overt or covert.

The linearization algorithm I will propose is based on paths. As we've seen, framing Contiguity in terms of paths in the way that (34) does leaves its effects unchanged for phrase markers that don't have multidominance in them, but has useful effects in situations where multidominance arises. The role that asymmetric c-commanding phrases have in the MLCA will be taken up by paths in my algorithm. Words will get into a linearization by virtue of the paths they have, and so I will state Totality in terms of paths too. This will also allow a phrase marker that has multidominance, and therefore more than one path for a word or group of words, to satisfy Totality by choosing just one of those paths. Finally, because the formalism for representing linearizations is a set of ordered pairs, (34) will have to be expressed in a way that references those ordered pairs rather than the strings they correspond to. Here, then, is a system that does those things.[8]

(39) Path Correspondence Algorithm (PCA)

a. Let $p(w)$=(XP$_1$, XP$_2$,..., XP$_n$), a path, be the set of phrases that dominate w, a word, and include the root phrase such that every XP$_i$ is dominated by every XP$_{j \leq i}$.

b. $\Pi(P)$ is a set of paths formed from the words in P.

c. d(XP) is the set of ws such that XP is in $p(w)$. d(w) is w.

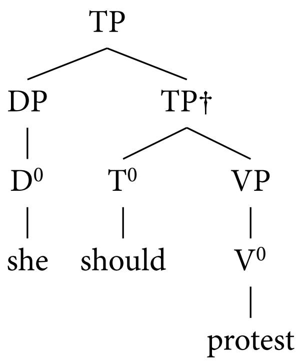
d. If p, a path, is in Π, then for every $XP \in p$, either $a < b$ or $b < a$ is in the linearization, for all $a \in d$(XP) and $b \in d(\beta)$, β XP's sister.

e. Totality
For every w in P, $\Pi(P)$ must contain $p(w)$.

Totality requires that every word in a sentence be associated with a path that is used to linearize it. The sum of these paths is Π. For each of these paths, (39d) then introduces Contiguity-preserving ordered pairs into the linearization. (39d) doesn't make the language particular correct choices – that must come from a part of the linearization scheme that fixes the choices among the cross-linguistic word-orders – but it limits those choices to just ones that satisfy Contiguity.

8 Note that the PCA does not need Antisymmetry to derive Terseness. It follows from the part of the PCA that enforces contiguity. Indeed, it could be that the PCA derives Antisymmetry.

We'll look at two case studies to see how the PCA does its job. Consider first a vanilla phrase-marker with no multidominance.

(40)


For each of the words in (40), there is only one path. Consequently, the smallest Π that satisfies Totality is (41).

(41) a. *p*(*she*) = {DP, TP}
b. *p*(*should*) = {TP†, TP}
c. *p*(*protest*) = {VP, TP†, TP}

From these paths, we can calculate *d*, which relates phrases to the words that are linearized by (39d). The *d* of a phrase are all the words that contain that phrase in its path.

(42) a. *d*(TP) = {*she*, *should*, *protest*}
b. *d*(DP) = {*she*}
c. *d*(TP†) = {*should*, *protest*}
d. *d*(VP) = {*protest*}

(39d) requires that each of the sets in (42) map onto a contiguous substring in the linearization. For instance, for (39d) to hold of TP†, all of the words in *d*(TP†) (i.e., *should* and *protest*) must be ordered in the same way to the words in TP†'s sister: *d*(DP) (i.e., *she*). Every phrase that is in some word's path will be subject to this requirement, and so every word will be part of a series of phrases that are contiguous, each larger phrase in that path mapping onto a a larger contiguous superstring containing that word.

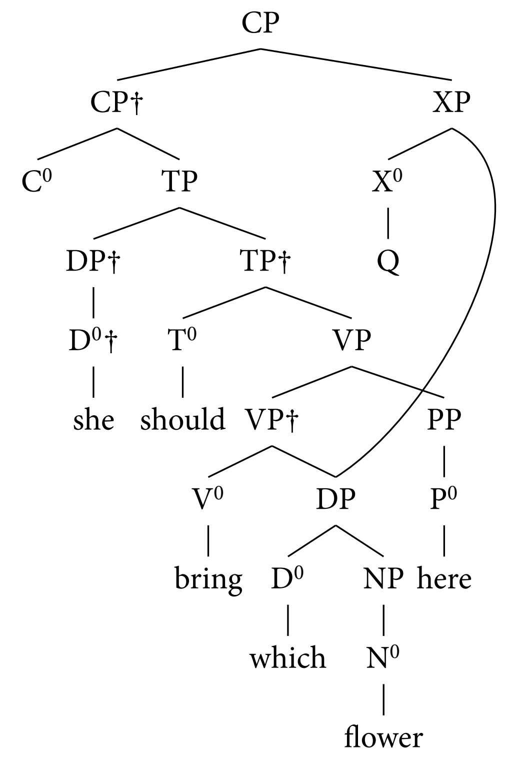
The PCA therefore allows for the linearizations of (35) in (43).

(43) a. she should protest
b. should protest she
c. she protest should
d. protest should she

This is probably more possibilities than should be allowed – (43d) is a sufficiently rare way for a language to linearize this structure that we might want to block it – but it comes close to what's cross-linguistically available. I will assume that the language particular choices narrow this set down to the particular outcomes appropriate for any particular language. English (a head initial, Specifier initial language) chooses (43a).

The second case study is (35).

(35)



As we've seen, *which* and *flower* have two paths in (35), and so the largest Π contains them both:

(44) a. p(*which*) = {DP, VP†, VP, TP†, TP, CP†, CP}
b. p(*which*) = {DP, XP, CP}
c. p(*flower*) = {NP, DP, VP†, VP, TP†, TP, CP†, CP}
d. p(*flower*) = {NP, DP, XP, CP}
e. p(*bring*) = {VP†, VP, TP†, TP, CP†, CP}
f. p(*here*) = {PP, VP, TP†, TP, CP†, CP}
g. p(*should*) = {TP†, TP, CP†, CP}
h. p(*she*) = {DP†, TP, CP†, CP}
i. p(*Q*) = {XP, CP}

The values for *d* are:

(45) a. *d*(CP) = {*she*, *should*, *bring*, *here*, *Q*, *which*, *flower*}
  b. *d*(XP) = {*Q*, *which*, *flower*}
  c. *d*(CP†) = {*she*, *should*, *bring*, *here*, *which*, *flower*}
  d. *d*(TP) = {*she*, *should*, *bring*, *here*, *which*, *flower*}
  e. *d*(DP†) = {*she*}
  f. *d*(TP†) = {*should*, *bring*, *here*, *which*, *flower*}
  g. *d*(VP) = {*bring*, *here*, *which*, *flower*}
  h. *d*(VP†) = {*bring*, *which*, *flower*}
  i. *d*(DP) = {*which*, *flower*}
  j. *d*(NP) = {*flower*}

(39d) prevents almost all linearizations of (44). It allows a linearization for this Π only under very narrow circumstances: when the language's word order settings would allow the multidominant phrase to be simultaneously contiguous to the sisters it has in both of its positions. Because of (45b), (39d) requires the linearization to have a contiguous string made from *Q, which* and *flower*. But because of (45g) and (45h), it also requires contiguous substrings made from {*bring, which, flower*} and {*bring, which, flower, here*}, which means the linearization must have one of the strings in (46) in it.

(46) a. i. bring which flower here
    ii. bring flower which here
  b. i. here bring which flower
    ii. here bring flower which
  c. i. which flower bring here
    ii. flower which bring here

The strings in (46a) can't coexist in a linearization that also puts *Q* contiguous with {*which, flower*}. The strings in (46b) and (46c) can if nothing in larger phrases separates Q. For instance, the strings in (47) would satisfy (39d).

(47) a. Q which flower bring here should she
  b. she should here bring which flower Q

I don't know of such a case, but I don't know of any harm in letting in this possibility. In general, though, (44) is too large to have a viable outcome. A smaller Π will have to be chosen.

There are four other Πs that satisfy Totality. They all give to *which* and *flower* just one path. One such Π chooses paths for *which* and *flower* that go through XP;

another chooses paths for *which* and *flower* that go through VP† instead. The first of these is (48) and the second (49).

(48) a. *p*(*which*) = {DP, XP, CP}
  b. *p*(*flower*) = {NP, DP, XP, CP}
  c. *p*(*bring*) = {VP†, VP, TP†, TP, CP†, CP}
  d. *p*(*here*) = {PP, VP, TP†, TP, CP†, CP}
  e. *p*(*should*) = {TP†, TP, CP†, CP}
  f. *p*(*she*) = {DP†, TP, CP†, CP}
  g. *p*(*Q*) = {XP, CP}

(49) a. *p*(*which*) = {DP, VP†, VP, TP†, TP, CP†, CP}
  b. *p*(*flower*) = {NP, DP, VP†, VP, TP†, TP, CP†, CP}
  c. *p*(*bring*) = {VP†, VP, TP†, TP, CP†, CP}
  d. *p*(*here*) = {PP, VP, TP†, TP, CP†, CP}
  e. *p*(*should*) = {TP†, TP, CP†, CP}
  f. *p*(*she*) = {DP†, TP, CP†, CP}
  g. *p*(Q) = {XP, CP}

The *d*s for (48) are in (50), and they correspond to the string in (51) in a head-initial and Specifier-initial language like English.

(50) a. *d*(CP) = {*she*, *should*, *bring*, *here*, *Q*, *which*, *flower*}
  b. *d*(XP) = {*Q*, *which*, *flower*}
  c. *d*(CP†) = {*she*, *should*, *bring*, *here*}
  d. *d*(TP) = {*she*, *should*, *bring*, *here*}
  e. *d*(DP†) = {*she*}
  f. *d*(TP†) = {*should*, *bring*, *here*}
  g. *d*(VP) = {*bring*, *here*}
  h. *d*(VP†) = {*bring*}
  i. *d*(DP) = {*which*, *flower*}
  j. *d*(NP) = {*flower*}

(51) Q which flower she should bring here

The *d*s for (49) are in (52), and they correspond to the string in (53), in a head-initial, Specifier-initial language.

(52) a. *d*(CP) = {*she*, *should*, *bring*, *here*, *Q*, *which*, *flower*}
  b. *d*(XP) = {*Q*}

c. *d*(CP†) = {*she*, *should*, *bring*, *here*, *which*, *flower*}
d. *d*(TP) = {*she*, *should*, *bring*, *here*, *which*, *flower*}
e. *d*(DP†) = {*she*}
f. *d*(TP†) = {*should*, *bring*, *here*, *which*, *flower*}
g. *d*(VP) = {*bring*, *here*, *which*, *flower*}
h. *d*(VP†) = {*bring*, *which*, *flower*}
i. *d*(DP) = {*which*, *flower*}
j. *d*(NP) = {*flower*}

(53) Q she should bring which flower

These are the desired outcomes; they correspond to the overt and covert movement possibilities.

The remaining two Πs that satisfy Totality give to *which* and *flower* divergent paths. They are both blocked by the PCA. To see how, consider (54), where *flower* is given a path through XP and *which* is given a path through VP†.

(54) a. *p*(*which*) = {DP, VP†, VP, TP†, TP, CP†, CP}
b. *p*(*flower*) = {NP, DP, XP, CP}
c. *p*(*bring*) = {VP†, VP, TP†, TP, CP†, CP}
d. *p*(*here*) = {PP, VP, TP†, TP, CP†, CP}
e. *p*(*should*) = {TP†, TP, CP†, CP}
f. *p*(*she*) = {DP†, TP, CP†, CP}
g. *p*(Q) = {XP, CP}

The *d*s for (54) are (55).

(55) a. *d*(CP) = {*she*, *should*, *bring*, *here*, *Q*, *which*, *flower*}
b. *d*(XP) = {*Q*, *flower*}
c. *d*(CP†) = {*she*, *should*, *bring*, *here*, *which*}
d. *d*(TP) = {*she*, *should*, *bring*, *here*, *which*}
e. *d*(DP†) = {*she*}
f. *d*(TP†) = {*should*, *bring*, *here*, *which*}
g. *d*(VP) = {*bring*, *here*, *which*}
h. *d*(VP†) = {*bring*, *which*}
i. *d*(DP) = {*which*, *flower*}
j. *d*(NP) = {*flower*}

d(VP†) and d(VP) together require that the linearization produce the string *bring which here* (once English-specific choices are made). But d(DP) requires that the linearization also produce the string *which flower*. There is no way of linearizing these words that preserves these two requirements. Exactly the same incompatibility arises if the path for *flower* goes through VP† and the path for *which* goes through XP – the other way of choosing divergent paths for these words. The reason these choices lead to a conflict is because all choices of paths for *which* and *flower* will contain DP, and (39d) will consequently require *which* and *flower* to be contiguous. This is how this system prevents the words in a moved phrase from getting linearized in different positions.

The PCA, then, allows for both overt and covert movement and, like the MLCA, explains why multidominant structures allow for selective relaxation of Contiguity. It makes Contiguity, rather than asymmetric c-command, the driving force behind a linearization. The formalization of Contiguity involved enforces a particular kind of "nesting" condition on entire phrase markers. It allows multidominance in just those cases where that nesting condition can be satisfied for every word in the phrase marker without considering the complete structure of the sentence.

4 Summary

What I've shown here is a way of completing Nunes' method of deriving Terseness that involves defining the "copy of α" as "giving α an addition position in the phrase marker." Traditional linearization schemes have stood in the way of such a move. I've offered two new linearization algorithms that don't, each with slightly different empirical footprints.

References

Abels, Klaus, and Ad Neeleman. 2012. Linear asymmetries and the LCA. *Syntax* 15:25–74.

Engdahl, Elisabet. 1980. The syntax and semantics of questions in Swedish. Doctoral Dissertation, University of Massachusetts, Amherst, Massachusetts.

Engdahl, Elisabet. 1986. *Constituent questions*. Dordrecht, The Netherlands: D. Reidel Publishing Company.

Gärtner, Hans-Martin. 2002. *Generalized transformations and beyond*. Berlin: Akademie-Verlag.

Johnson, Kyle. 2012. Toward deriving differences in how *Wh* movement and QR are pronounced. *Lingua* 122:529–553.

Kayne, Richard S. 1994. *The antisymmetry of syntax*. Cambridge, Massachusetts: MIT Press.

Nunes, Jairo. 1995. The copy theory of movement and linearization of chains in the Minimalist Program. Doctoral Dissertation, University of Maryland.

Nunes, Jairo. 1996. On why traces cannot be phonetically realized. In *Proceedings of North East Linguistic Society*, ed. Kiyomi Kusumoto, 211–226. Harvard University and MIT: Graduate Linguistic Student Association.

Nunes, Jairo. 1999. Linearization of chains and phonetic realization of chain links. In *Working minimalism*, ed. Samuel Epstein and Norbert Hornstein, 217–249. Cambridge, Massachusetts: MIT Press.

Nunes, Jairo. 2004. *Linearization of chains and sideward movement*. Linguistic Inquiry Monographs. Cambridge, Massachusetts: MIT Press.

Peters, Stanley, and Robert W. Ritchie. 1981. Phrase linking grammars: Draft only. Unpublished paper, Stanford University, December 1981.

Selkirk, Elisabeth. 2011. The syntax-phonology interface. In *The handbook of phonological theory*, ed. John Goldsmith, Jason Riggle, and Alan C. Yu, 435–484. Malden, MA: Wiley-Blackwell, 2 edition.

Starke, Michal. 2001. Move dissolves into merge: A theory of locality. Doctoral Dissertation, University of Geneva.

de Vries, Mark. 2007. Internal and external remerge: On movement, multidominance, and the linearization of syntactic objects. Unpublished manuscript, University of Groningen.