Locality and the distribution of features

Elise Newman (based on joint work with Kenyon Branan)
elise.newman@ed.ac.uk

April 5, 2023

1 Some puzzles about locality

- Theories of locality ideally tell us what things are accessible to various kinds of dependencies.

(1) [X ... [... [... [... Y [... [...
Can Y move to X?
Can X agree with Y?
Can X control Y?
etc.

- We know that dependency formation is impacted by several factors...

1. Whether there is an intervener...

(2) [X ... [... [... Y_1☺ [... Y_2☹ [... [...
Y_2 can't be a dependent of X because Y_1 is closer.

(3) Superiority effects:
a. Who bought what?
b. *What did who buy?

2. Whether the dependent is embedded in a clause, and if so, what kind...

(a) Complement clauses trigger successive cyclicity.

(4) [X ... [... [$_{CP}$ C ... [... [... Y [... [...
Y can't be a dependent of X unless it moves to Spec CP first.

(5) West Ulster *all*-stranding in embedded Spec CP (McCloskey, 2000, ex.8,9)
a. What did he say [<what> all (that) he wanted <what all>]?
b. Where do you think [<where> all they'll want to visit <where all>]?

(b) Adjunct clauses often block dependencies across them entirely.

(6) [X ... [... [$_{AdjP}$ ... [... [... Y☹ [... [...
Y can't be a dependent of X.

(7) *What did Sue arrive [before Max ate <what>]?

3. Whether the dependent is embedded in a specifier.

(8) [X ... [... [... [[$_{spec}$ ... Y☹ ...] ... [... [...
Y can’t be a dependent of X.

(9) *What did [pictures of <what>] impress Sue?

- This is a lot of different-looking conditions... it would be nice if we could appeal to some more general theory, that captures all of the above.

- **Today**:
  - We will explore some assumptions about Merge, and see where they take us.
  - They’ll take us to a theory of feature projection, which gives us a hook into a picture of locality.
  - We will adopt a toy theory of feature projection based on the CED...

    (10) *The Condition on Extraction Domains* (CED) (Huang, 1982; Chomsky, 1986; Cinque, 1990; Manzini, 1992):
    Movement may not cross a barrier XP, unless XP is a complement.

  - ...and show how it derives successive cyclicity without appealing to phase theory.

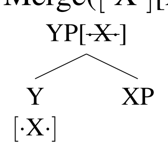
---

2 Some thoughts on feature-driven Merge

- An assumption from Chomsky (1995): *Merge is feature-driven*
  - Features that drive Merge get *checked* when they merge with certain other features.
  - Suppose that the checking algorithm looks like the function in (12).

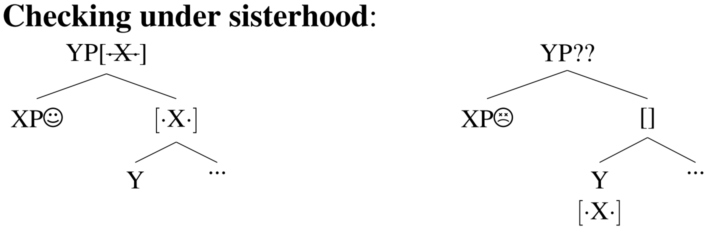
(11) [$\cdot\alpha\cdot$] = an instruction to merge with a bearer of α (Müller, 2010)

(12) Merge([·X·][X]) = [~~·X·~~]



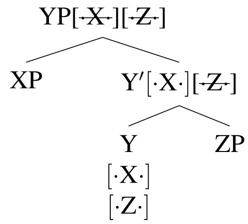
- A second assumption: *feature-checking takes place under sisterhood* (Neeleman & van de Koot, 2002; Adger, 2003; Zeijlstra, 2020)

(13) **Checking under sisterhood**:



- What does this do for us?
  - It requires us to have a theory of feature projection, or else we couldn't have specifiers.

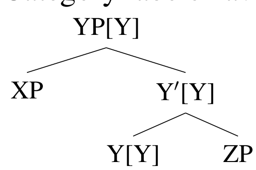
    (14) Merge features originate on heads, and project to bar-level nodes to license specifiers

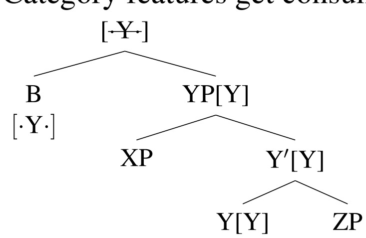
  - We have checked features floating around the tree (see Chomsky 1995, page 256 for some precedent).
    * Adger (2003) says that those features get deleted, but then we need an extra operation *Delete* applying at every instance of Merge... what if we just let checked features stick around?

- Ok, so we have features like [X], [·X·], and [~~·X·~~], and we need some algorithm deciding where, when and which of them get projected to have a complete picture of clause construction.
  - Received wisdom:
    1. Category features (e.g. [X]): must project at least to the maximal node for labeling purposes, probably not further if they get selected

       (15) Category labels have features of their heads

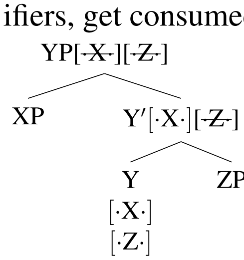
       

       (16) Category features get consumed by selection

    2. Unchecked features (e.g. [·X·]): must project at least to the bar-level node to license specifiers

       (14) Merge features originate on heads, and project to bar-level nodes to license specifiers, get consumed by the time the maximal node is built

3. Checked features (e.g. [~~X~~]): what I want to figure out today.

- Why do I care about the projection of checked features?
  - Because they encode information about the make-up of a constituent.

(17) What these features mean

a. [X] = I am an X

b. [·X·] = I want an X

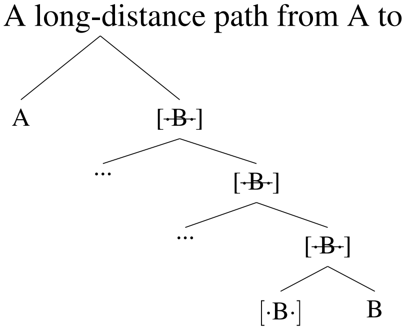
c. [~~X~~] = I am not an X, but I dominate one

∗ What if higher probes used this information to determine whether and what kinds of goals were accessible to them?

(18) **Accessibility**
A probe A searching for a goal B may only initiate Search for B if there is a path from A to B.
(taking inspiration from Kayne 1981; Pesetsky 1982; McFadden & Sundaresan 2019)

(19) **Path**
There is a Path from A to B if every node that dominates B, up to A's sister, bears a feature checked by B.

(20) A long-distance path from A to B



∗ Paths facilitate dependencies by guiding the probe's Search for its goal.

· Chomsky (2004) suggests that *probing* involves an operation of "Minimal Search" → the search procedure stops once a match has been found

See Branan & Erlewine (2021) for an overview of possible Search procedures and Preminger 2019; Ke 2019; Atlamaz 2019; Krivochen 2022; Chow 2022 for more specific proposals.

· On this view, Search isn't blind: it only examines nodes bearing [~~B~~] (until it finds a B-bearing element)

· If A's sister does not bear a feature checked by B (i.e. because there is no local B), Search fails at the outset, without examining any nodes in the tree.

- **Takeaway**:
  - Feature-driven Merge + checking under sisterhood implies projection of features, including checked features.

- Projected checked features created paths between probes and goals.

- **Next**:
  - Developing conditions on feature projection

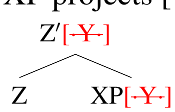
2.1 Towards a theory of feature projection

- So far we have two ingredients of the theory:
  1. Ingredient 1: checked features stick around
  2. Ingredient 2: some amount of feature projection happens
- Question: under what general conditions do features project?
  - The most ideal theory would make as few distinctions as possible
    * "Everything projects" or "nothing projects" is better than "some things but not others project"
      · We know that at least unchecked features and category features project sometimes, so we can't say nothing projects.
      · Conclusion: "everything projects" is the next best option → checked features project
- The CED says non-complements are opaque for movement → maybe this is due to the feature projection algorithm.

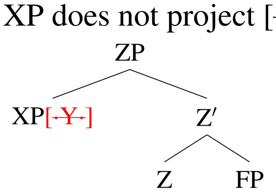
  (21) **A feature projection algorithm**:
  a. Non-maximal projections project all of their features to the maximal node.
  b. Complements (i.e. sisters of heads) project all of their features.
  c. Specifiers/adjuncts project no features.

- We suggest a more nuanced version of the feature projection algorithm in Newman & Branan (2023), which accounts for both the CED and its exceptions. This simpler one will do for now.
- Illustrating feature projection from maximal projections in different contexts.

(22) XP projects [~~Y~~] to a higher node if it is a complement



(23) XP does not project [~~Y~~] to a higher node if it is a specifier



- **Takeaway**:
  - Features may project past maximal projections if they are complements.
- **Next**:
  - Showing how this lays the groundwork for a unified theory of locality

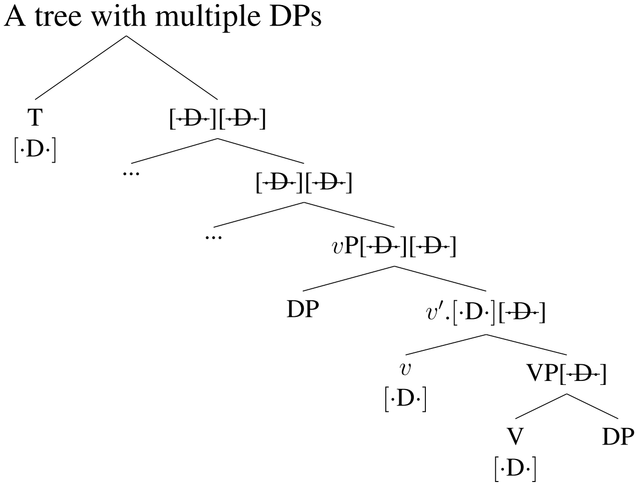
3 Extending the theory

- We have a theory that predicts complements to be transparent while specifiers/adjuncts are opaque.
  - Digging a bit deeper, what do the relevant paths actually look like?
  - And can this approach also explain the other kinds of locality effects that we discussed, rather than just the CED?
- Exciting result: the theory subsumes Relativized Minimality and Phase Theory.

3.1 Intervention effects

- A question arises: if there are two DPs selected in some structure, are multiple [·~~D~~·] features projected?
  - If so, does this matter?

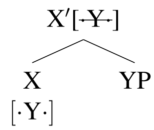
(24) A tree with multiple DPs



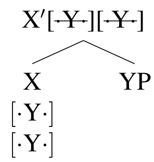
  - Recall: features represent types rather than tokens
    - ∗ Type notions of features are necessary if multiple checking is possible.

(25) Multiple features checked by the same element at the same time: (a) and (b) have the same structures/derivations but project different features

a. One checked Y feature

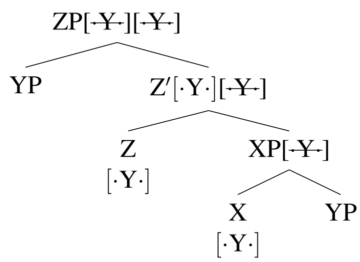


b. Two checked Y features

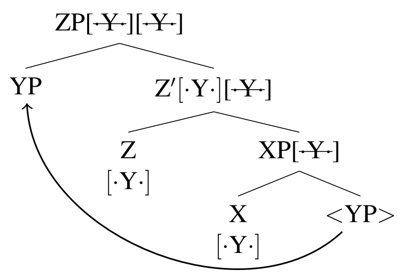


(26) Multiple features checked by different elements at different times: (a) and (b) have different structures/derivations but project the same features

a. Two checked Y features = two merged YPs



b. Two checked Y features = one merged+re-merged YP



- Conclusion: number of checked features on a node does not uniquely correspond to the number of distinct checkers

- In other words, having a checked Y feature means something like *has the property of containing at least one YP*.

3.1.1 Redundancy

- With type notions of features, the above trees have a lot of redundancy.

(27) Multiple checked features are redundant
XP[~~·Y·~~][~~·Y·~~] contains the same information as XP[~~·Y·~~]

- Suppose: the derivation doesn't want to carry around redundant information, in which case let's imagine that instances of XP[~~·Y·~~][~~·Y·~~] always get converted into XP[~~·Y·~~].
  - Prediction 1: multiple instances of the same features are not allowed

- Prediction 2: if two features stand in an entailment relationship, only the more specific one should project

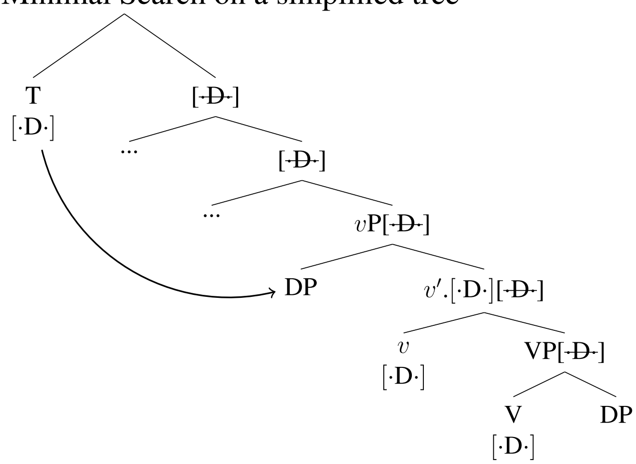
(28) Predictions schematized:

a. Prediction 1: *XP[~~·Y·~~][~~·Y·~~], ✓XP[~~·Y·~~]

b. Prediction 2: *XP[~~·Y·~~][~~·Z·~~], where Y→Z, ✓XP[~~·Y·~~]

- Now putting it all together: Intervention effects just follow from Minimal Search
  - I won't give an explicit Search algorithm here, but see Branan & Erlewine (2021) for some ideas.
  - The gist: the probe searches every node with [~~·D·~~] until it finds a DP, and then stops.

(29) Minimal Search on a simplified tree



- **Takeaway**:
  - Features are types rather than tokens.
  - Redundant features get deleted.
  - Minimal Search is the only locality principle we need to capture CED and intervention effects.
- **Next**:
  - Looking at paths of unselected features, which behave surprisingly, and showing how constraining those paths enforces successive cyclic movement → covers the facts that phase theory was developed for

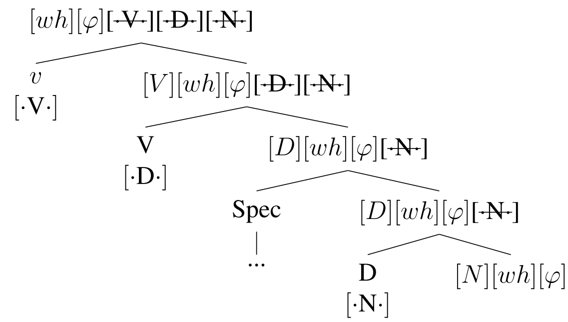
3.2 Thinking about non-category features

- I've said that projection from a maximal projection makes it transparent to dependencies across it.

- That's *technically* true. But that doesn't necessarily mean it is transparent to *movement*.
  - If movement is another kind of feature checking (like all Merge), the right features have to get projected, or else checking will occur in situ.

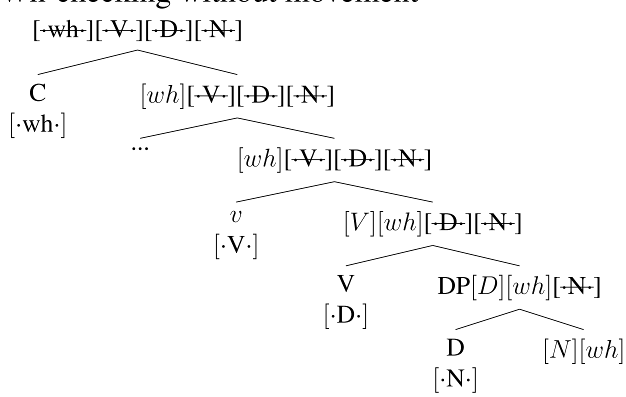
- We saw that category features get consumed by selection, to produce checked versions of those features
- From that perspective, features not involved in selection should project indefinitely, like checked features, and crucially only from complements.

(30) Projection of non-selected features



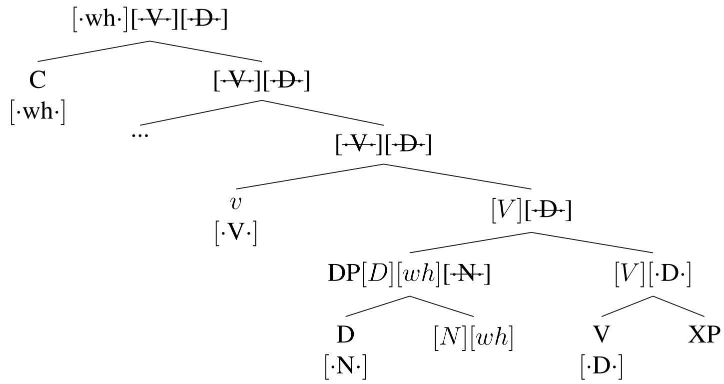
- Prediction 1: by default, feature-checking operations involving non-category features occur at a distance

(31) Wh-checking without movement



- Prediction 2: only wh-features on complements are accessible to wh-checking. The projection algorithm prevents the features of specifiers from projecting.

(32) No Wh-checking by specifiers



- In what follows, I'll show that the distribution of [·wh·] is predictable from the feature projection algorithm → captures successive cyclic movement

3.2.1 Wh-movement

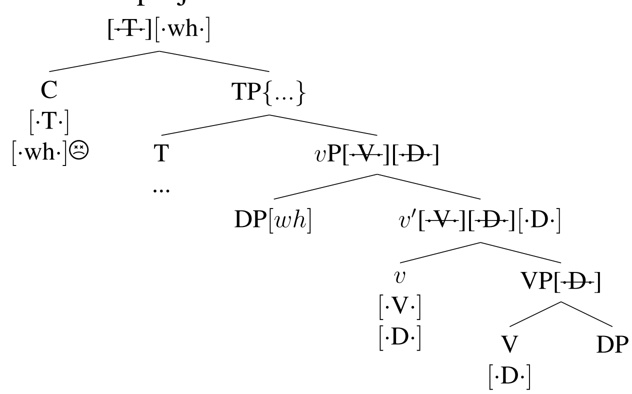
- Two puzzles

  1. Why aren't all languages wh-in-situ?
  2. How do specifiers ever control wh-dependencies?

- Proposal: the answer to the second question answers the first, and gives us clause-medial successive cyclicity in the process.

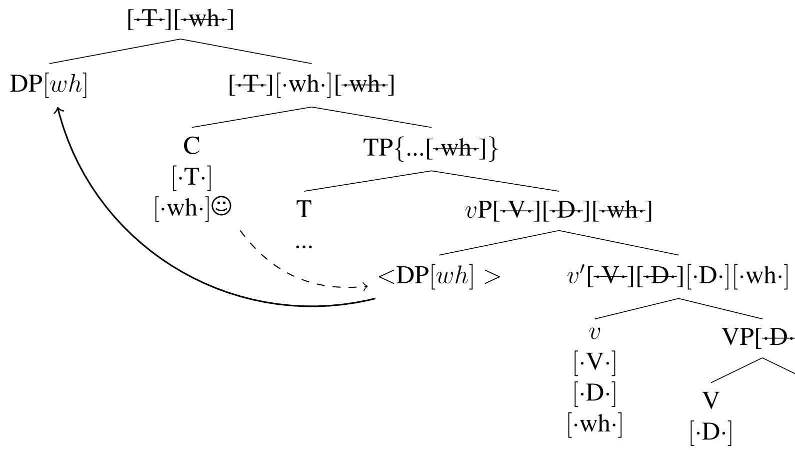
  – Illustrating the problem: first specifiers don't project their features – there is therefore no path to them, their features can't be used to check [·wh·] in situ either.

    (33) If C is looking for wh-features on the subject, it can't find them because [~~-wh-~~] is not one of the projected features on C's sister



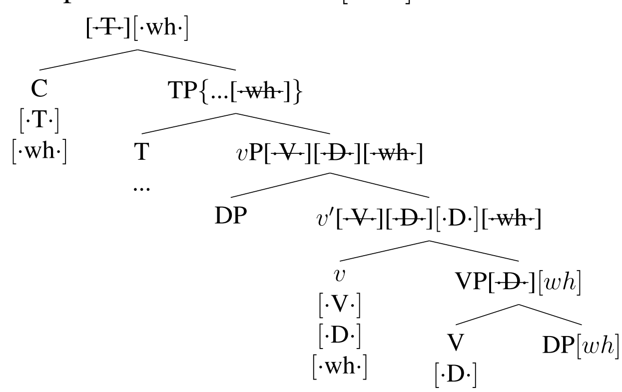
  – In order for C to check its [·wh·] feature, the subject must check some other [·wh·] earlier in the derivation, i.e. upon first merge or when it moves to Spec TP (in languages with that kind of subject movement).

    (34) If the subject checks a [·wh·] in situ, a path of [~~-wh-~~] gets projected to C, leading it to attract the subject.



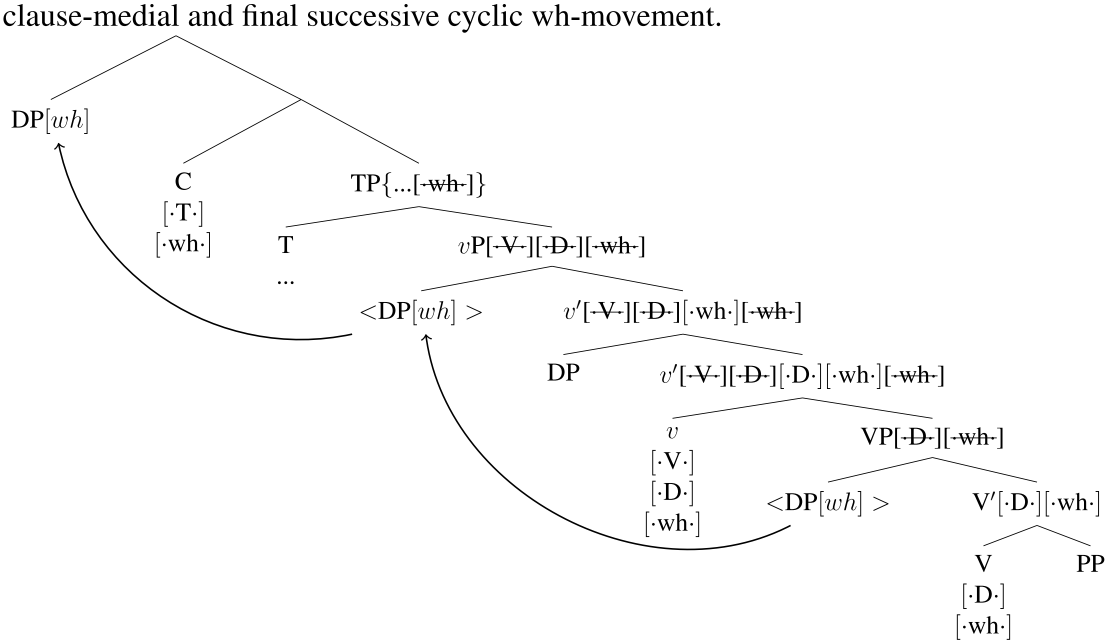
– Two birds with one stone: if a language has a [·wh·] on v, wh-subjects are not only accessible for wh-movement, but wh-objects are inaccessible for wh-in-situ.

(35) Complements can't check [·wh·] on C at a distance if v has [·wh·]



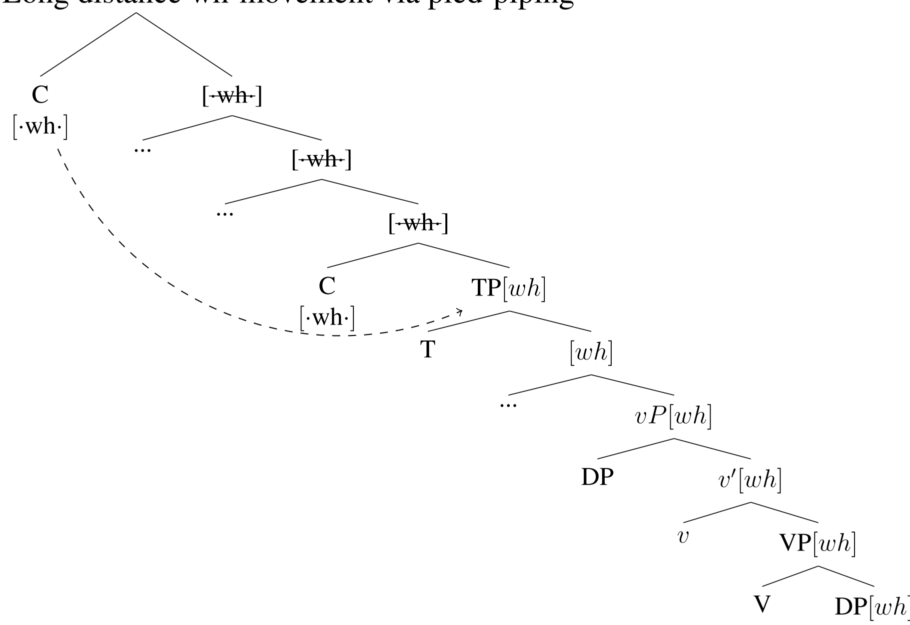
- Side effect: we probably need V to have [·wh·] as well, so direct object specifiers can also wh-move.

(36) Objects check a [·wh·] in situ as well, projecting a path of [~~·wh·~~] to v and C, leading to clause-medial and final successive cyclic wh-movement.



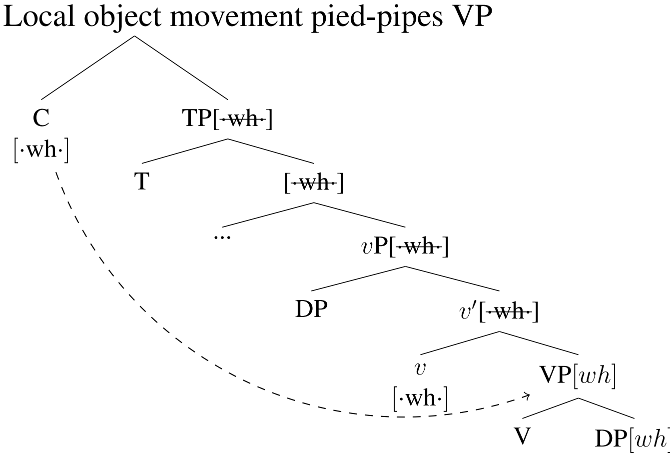
- In sum: having [·wh·] on argument introducing heads blocks wh-in-situ, and makes all arguments accessible for wh-movement.
- Parametric variation: Not having [·wh·] on argument introducing heads leads to wh-checking at a distance (i.e. wh-in-situ), and only by complements.
  - Some fun pied-piping predictions:
    - If a language only has [·wh·] on C heads, it should look wh-in-situ within a single clause, but long distance object movement should trigger pied-piping of TP.

(37) Long distance wh-movement via pied-piping



    - If a language only has [·wh·] on C and v, but not V, then subjects will wh-move like usual, but objects will only move if they are complements, and they will pied-pipe VP.

(38) Local object movement pied-pipes VP



  - Lots of puzzles to work out, such as why is pied-piping ever optional?

(39) Optional pied-piping of P
  a. Who did Sue buy a picture of?
  b. Of whom did Sue buy a picture?

- In this system, the only way to get wh-movement of specifiers is to have a particular distribution of [·wh·] features on heads.
  - These features make wh-specifiers visible to higher probes, but also trigger successive cyclic wh-movement of wh-complements.
  - The system therefore offers a partial solution to the question of why wh-movement is successive cyclic through various positions – it's a prerequisite for having wh-movement of particular arguments.
- Why is wh-movement successive cyclic through the edge of CP? (you might ask)
  - I think this problem has a boring answer:
    - For wh-movement to matrix Spec CP to be possible, C must have [·wh·].
    - If all C heads have the same syntactic features, then every C has [·wh·] $\rightarrow$ wh-movement travels through the edge of every CP.
    - This feature can unproblematically go unchecked in the absence of a wh-phrase (Preminger, 2014; Longenbaugh, 2019).

- **Takeaway**:
  - The distribution of [·wh·] tells us what positions wh-movement must target.
  - The kinds of elements that are permitted to wh-move heavily constrain the distribution of [·wh·].
  - Phase theory: the result of these pressures on the distribution of [·wh·]

4 Conclusion

- **The logic of the approach**:
  - Every instance of Merge takes two sets of features as input and projects some set of features as an output.
  - Two things tell us about the output
    1. An [·X·] licensing Merge suppresses a corresponding [X] on its sister.
    2. If one of the sisters is a maximal projection, whether it projects depends on whether its sister is a head.
  - Projection of [~~X~~] creates paths, which guide probes to their goals via Minimal Search.
- **Main results**:
  - Intervention effects are as easy as ever.
  - Complements are always transparent; non-complements are opaque, yielding the CED.

    - Places constraints on the distribution of [·wh·], to license movement of specifiers $\rightarrow$ phase theory

- **Some puzzles and avenues**:
    - Optional pied-piping
    - φ-agreement?

References

Adger, David. 2003. *Core syntax*. Oxford: Oxford University Press.

Atlamaz, Ümit. 2019. *Agreement, case, and nominal licensing*: Rutgers University-School of Graduate Studies dissertation.

Branan, Kenyon & Michael Yoshitaka Erlewine. 2021. Locality and minimal search.

Chomsky, Noam. 1986. *Barriers*. MIT Press.

Chomsky, Noam. 1995. *The minimalist program*. Cambridge, MA: MIT Press.

Chomsky, Noam. 2004. Beyond explanatory adequacy. In Adriana Belletti (ed.), *Structures and beyond*, Oxford University Press.

Chow, Keng Ji. 2022. A novel algorithm for minimal search. *Snippets* (42).

Cinque, Guglielmo. 1990. *Types of Ā-dependencies*. Cambridge, MA: MIT Press.

Huang, C.-T. J. 1982. *Logical relations in Chinese and the theory of grammar*: MIT dissertation.

Kayne, Richard S. 1981. Ecp extensions. *Linguistic inquiry* 12(1). 93–133.

Ke, Hezao. 2019. *The syntax, semantics and processing of agreement and binding grammatical illusions*: dissertation.

Krivochen, Diego. 2022. The search for Minimal Search: A graph-theoretic view.

Longenbaugh, Nicholas. 2019. *On expletives and the agreement-movement correlation*. Cambridge, MA: MIT dissertation.

Manzini, Rita. 1992. *Locality, a theory and some of its empirical consequences*. Cambridge, MA: MIT Press.

McCloskey, James. 2000. Quantifier float and wh-movement in an Irish English. *Linguistic Inquiry* 31(1). 57–84.

McFadden, Thomas & Sandhya Sundaresan. 2019. Deriving selective opacity via path-based locality. Handout presented at Syntax Mini-course, University of Cambridge. `https://www.sndrsn.org/copy-of-agree-ment`.

Müller, G. 2010. On deriving CED effects from the PIC. *Linguistic Inquiry* 41(1). 35–82.

Neeleman, A. & H. van de Koot. 2002. The configurational matrix. *Linguistic Inquiry* 33(4). 529–574.

Newman, Elise & Kenyon Branan. 2023. Paths: the ghost of features past. Submitted.

Pesetsky, David Michael. 1982. *Paths and categories*: Massachusetts Institute of Technology dissertation.

Preminger, Omer. 2014. *Agreement and its failures*. The MIT Press.

Preminger, Omer. 2019. What the PCC tells us about 'abstract' agreement, head movement, and locality. *Glossa-an International Journal of Linguistics* 4(13). 1–42.

Zeijlstra, Hedde. 2020. Labeling, selection, and feature checking. In P. Smith, J. Mursell & K. Hartmann (eds.), *Agree to agree: Agreement in the minimalist programme*, 137–174. Berlin: Language Science Press.