3.2 Constituency (Merge)

The premise of Merge

- Language is recursive ← accepted fact
- Recurses on syntactic objects or constituents of a phrase. Empirically w/ constituency tests.

☒ We choose to represent recursive constituency as a tree structure: each node is a constituent

- How is a sentence formed?
  - → Bottom-up w/ tree grafting is Minimalism's answers.
  - → Why not top down? (Question for class.)
- The grafting of smaller syntactic objects into a larger one → Merge

Merge Constraints

- Clearly some applications of merge are ill-formed for the language {"I ate the apple was delicious." ; "The little house is" → "The house is little" ✗→ "House is the little"}
- Two approaches:

1) Filter now: never let an ill-formes syntactic object be formed, ie. restrict Merge's application

2) Filter later: let Merge apply to everything and let a future step take care of it.

Note: In some senses, the first may be simpler, reducing the search tree for a valid sentence; however, the machinery required for the second is less, provided that being ill-formed in merge ⇒ being ill-formed elsewhere.

- Stabler: First approach (à Chomsky 1980s)
- MCB: Second approach (Externalization)

★. Evidence from errors; which mistakes are made?

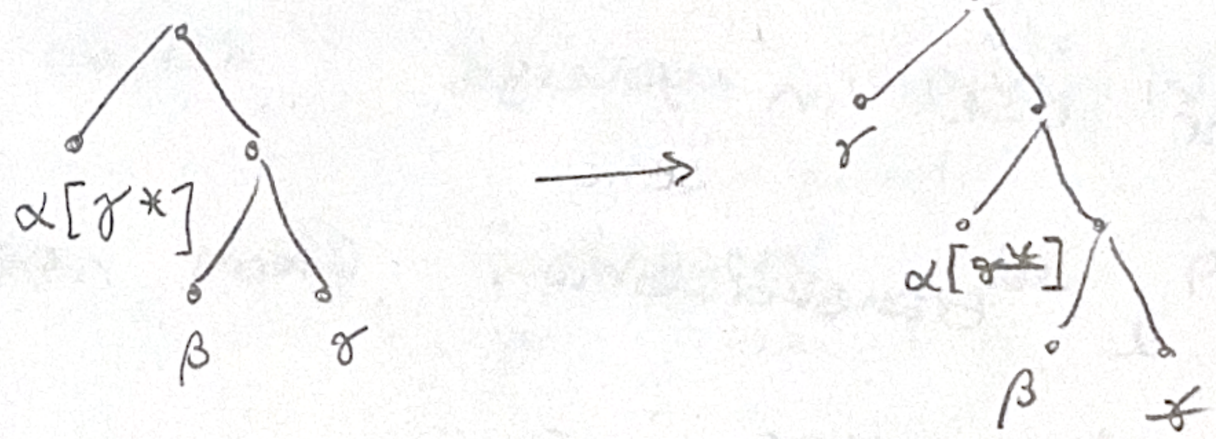
Internal Merge

- Both MCB and Stabler treat Move as a special type of Merge, called Internal Merge.

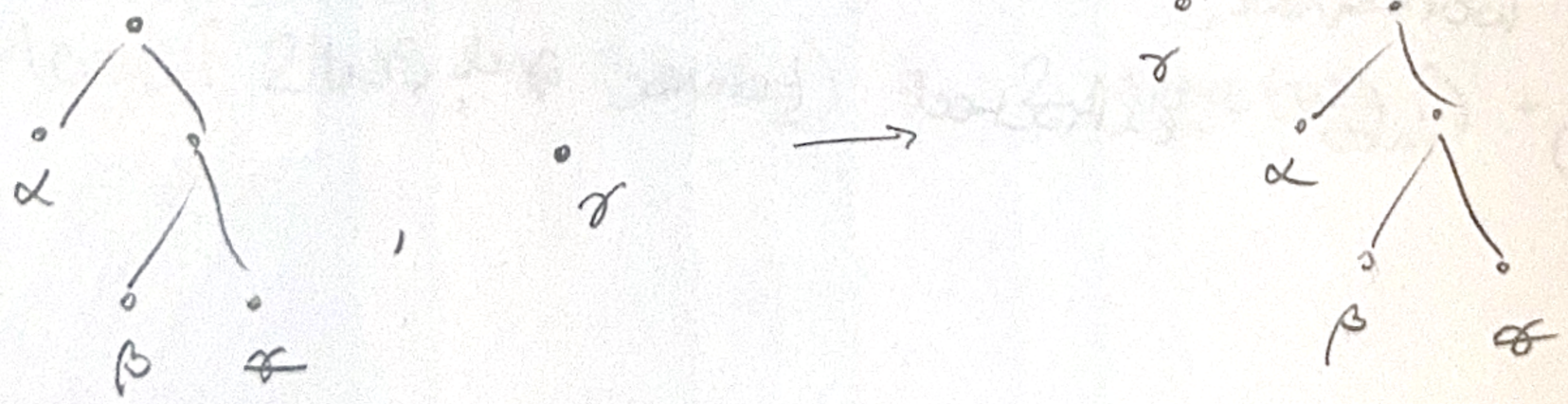
  → Stabler: Certain features trigger internal merge to be checked

  → MCB: Merge applies to all syntactic objects "extracted" from trees and their leftovers.

Stabler: Strong feature triggers



MCB: (only combination which survives externalization and optimality)



Activities

① Optimality is often criticized of assuming a perfect speaker. What mistakes would you expect speakers to make?

② Have students run through the derivation of simple sentences through Stabler and MCB's approach. Note any optimality or grammaticality assumptions made.

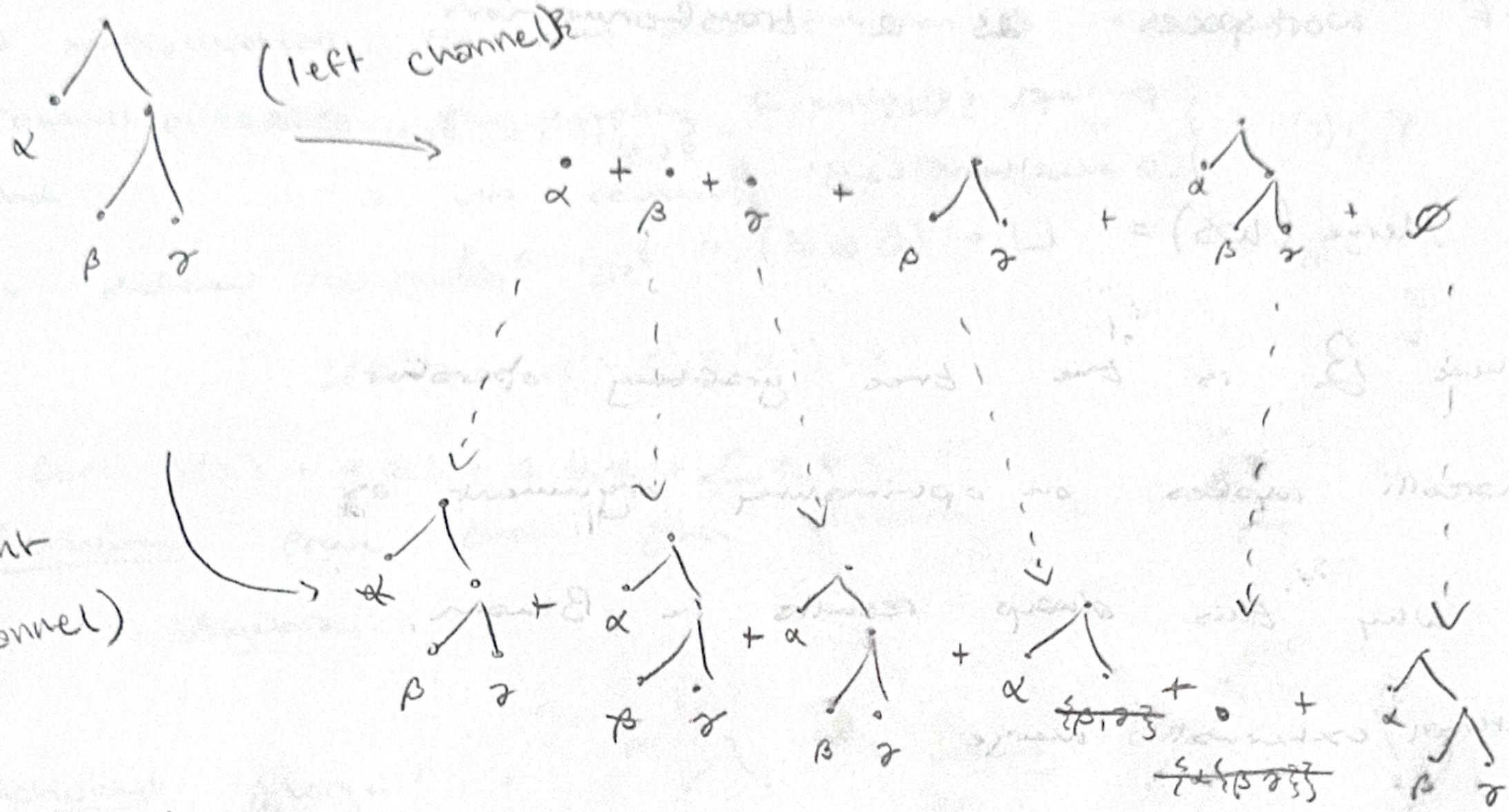
MCB's Merge: A closer look

- Workspaces are sets of syntactic terms available for use in Merge
- Merge is a "transformation" from one workspace to another.

→ If time: discuss FormSet & Select

- Given a workspace, we can decompose (comultiply) into extracted terms and leftovers.

Illustration:



We "add" the possibilities. If our input is multiple trees, we can extract only one Syntactic object. If our input is a linear combination of workspaces, we perform the operation on each WS and add/multiply.

$$\text{Merge}(T_1 \sqcup T_2) \longrightarrow \text{Merge}(T_1 \sqcup T_2)$$

$$\text{Merge}(2WS_1 + 3WS_2) = 2\,\text{Merge}(WS_1) + 3\,\text{Merge}(WS_2)$$

So merge is defined on the Vector space of workspaces as a transformation

$$\gamma_{S,S'}(F) = \begin{cases} F & F = S \sqcup S' \\ 0 & \text{otherwise} \end{cases}, \qquad \delta_{S,S'}(F) = \gamma_{S,S'} \otimes id$$

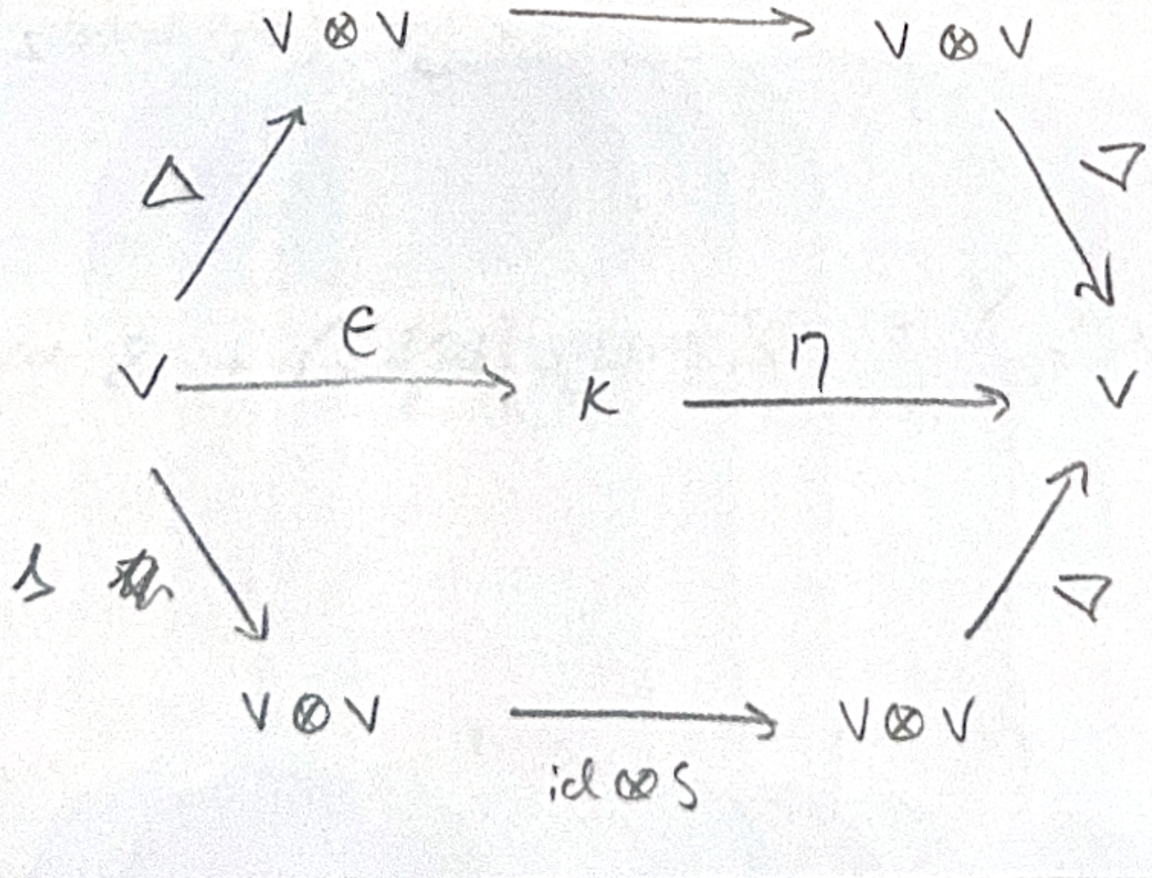
$$Merge_{S,S'}(WS) = \sqcup \circ (B \otimes id) \circ \delta_{S,S'} \circ \Delta$$

where B is the tree grafting operator.

Marcolli makes an optimality argument as to why this always results in Binary, internal/external merge.

Hopf Algebras

A Hopf Algebra is a vector space endowed with a coproduct Δ, a product ∇, a counit η, a unit ϵ, and an antipode S for which the following diagram commutes:



Exploring this definition.

MCB posits the extended disjoint union $\sqcup$ is a multiplication, the tree decomposition Δ is a comultiplication, the empty Forest is the unit, and is the counit; the antipode S is defined recursively

$$S(x) = -x - \sum S(x') \cdot x''$$

for $\Delta(x) = x \otimes 1 + 1 \otimes x + \sum x' \otimes x''$

Activities: prove that this defines a Hopf Algebra.

Activity: Merge • using this I like apples model.

Questions for future reading:

- Why are Hopf Algebras important?
- What benefits does this model pose over Computational Minimalism?
- What downsides?

Pair w/ discussion questions and LO's.