Lecture 4: From Formal Languages to Minimalism
Ma 191c: Mathematical Models of Generative Linguistics

Matilde Marcolli

Caltech, Spring 2024

consider again Context-free grammars $\mathcal{G} = (V_N, V_T, P, S)$

- V_N and V_T disjoint finite sets: *non-terminal* and *terminal* symbols
- $S \in V_N$ *start symbol*
- P finite rewriting system on $V_N \cup V_T$

$P =$ *production rules*: $A \to \alpha$ with $A \in V_N$ and $\alpha \in (V_N \cup V_T)^\star$

Language produced by a grammar $\mathcal{G}$:

$$\mathcal{L}_\mathcal{G} = \{w \in V_T^\star \mid S \overset{\bullet}{\to}_P w\}$$

language with alphabet V_T

Parse Trees of a context free language

• a finite, rooted, oriented (away from the root), planar tree (with a choice of a planar embedding)

• vertices decorated by elements of $V_N \cup V_T$ (terminal and non-terminal symbols)

• if an "internal vertex" (not a leaf) is decorated by A and if all the terminal vertices of oriented edges out of vertex A are labelled by $w_1, \ldots, w_n$ (with ordering specified by planar embedding) then

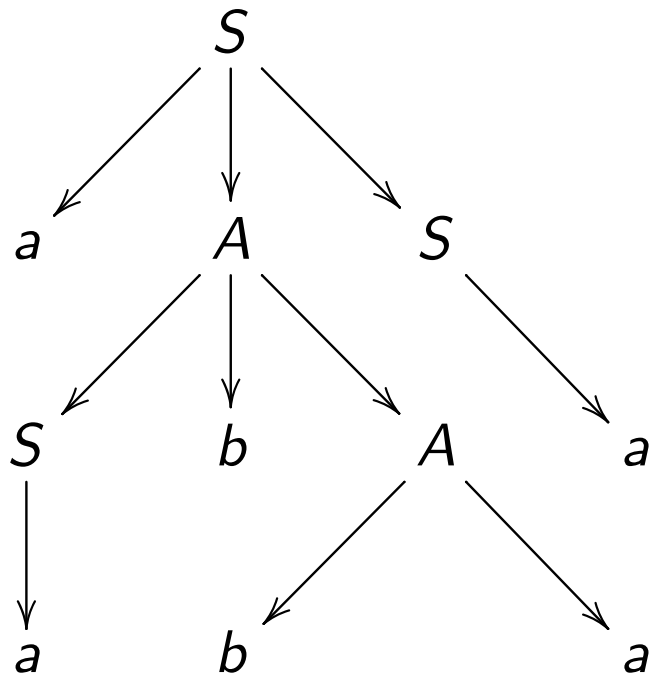
$$A \to w_1 \cdots w_n \quad \in P$$

Example

- Grammar: $\mathcal{G} = \{\{S, A\}, \{a, b\}, P, S\}$ with productions P

$$S \to aAS, \quad S \to a, \quad A \to SbA, \quad A \to SS, \quad A \to ba$$

- this is a possible parse tree for the string $aabbaa$ in $\mathcal{L}_{\mathcal{G}}$



Fact: for context-free $\mathcal{G} = (V_N, V_T, P, S)$ have a chain of derivations in $\mathcal{G}$

$$A \stackrel{\bullet}{\to} w_1 \cdots w_n$$

if and only if there is a parse tree for $\mathcal{G}$ with root decorated by A and with n leaves decorated by $w_1, \ldots, w_n$

to see this: if have parse tree with input A and outputs $w_1, \ldots, w_n$, show by induction on number of internal vertices that $A \stackrel{\bullet}{\to} w_1 \cdots w_n$ in $\mathcal{G}$

- if only root and leaves (no other vertices) then $A \to w_1 \cdots w_n$ is a production rule in P
- otherwise, assume know for all trees with $\leq k$ vertices (induction hypothesis); if tree has $k+1$ vertices, look at immediate successor vertices from root: get a production in P (from A to the list of successors) then for each successor that not leaf get a tree with $\leq k$ vertices

conversely if $A \overset{\bullet}{\to} w_1 \cdots w_n$ in $\mathcal{G}$

- then there is a chain of derivations in P,

$$A \to u_1, \quad \dots \quad u_i \to u_{i+1}, \quad \dots \quad u_k \to w_1 \cdots w_n$$

where the next derivation giving u_{i+1} is applied to some non-terminal element in the string u_i

- the first production rule $A \to u_1$, produces a string $u_1 = u_{11} \dots u_{1k_1}$ and gives a root labelled A with valence k_1 and leaves labelled by u_{1j}
- the second $u_1 \to u_2$ consists of some production rules in P applied to some of the non-terminal symbols u_{1j} in the string u_1: append trees to the vertices labelled u_{1j} with leaves the resulting strings in u_2
- continue with successive derivations until obtain a tree with root A and with leaves (ordered by planar embedding) labelled by $w_1, \dots, w_n$

Ambiguity of context-free languages (grammars)

- A context-free grammar $\mathcal{G}$ is *ambiguous* if there are words $w \in \mathcal{L}_{\mathcal{G}}$ that admit different (non-equivalent) parse trees
- Trivial example: $S \to A$, $S \to B$, $A \to a$, $B \to a$
- A language $\mathcal{L}$ is *inherently ambiguous* if every possible context-free grammar $\mathcal{G}$ with $\mathcal{L} = \mathcal{L}_{\mathcal{G}}$ is ambiguous

Example

$$\mathcal{L} = \{a^n b^n c^m d^m \mid n \geq 1, m \geq 1\} \cup \{a^n b^m c^m d^n \mid n \geq 1, m \geq 1\}$$

is inherently ambiguous because there are infinitely many strings of the form $a^n b^n c^n d^n$ that have different parse trees

Sketch of argument for Example:

Suppose $\exists$ unambiguous context-free $\mathcal{G}$ for $\mathcal{L}$ above: then can always arrange that for all $A \in V_N \setminus \{S\}$ have $A \overset{\bullet}{\to} x_1 A x_2$ with both x_1, x_2 not the empty word

• because of the form of words in $\mathcal{L}$ must have x_1 and x_2 consisting of only one type of symbol a, b, c, d (otherwise get a string not in $\mathcal{L}$)

• also symbol for x_1 different from symbol for x_2 (because of form of words cannot increase occurrences of only one type of symbol and still get words in $\mathcal{L}$) and also length of x_1 and x_2 has to be same (same reason)

• check only cases are x_1 made of a's and x_2 of b's or d's; x_1 made of b's and x_2 of c's; x_1 made of c's and x_2 of d's (divide variables other than S into C_{ab}, C_{ad}, C_{bc}, C_{cd})

• subdivide $\mathcal{G}$ into two grammars

$$\mathcal{G}_1 = \{\{S\} \cup C_{ab} \cup C_{cd}, V_T, P_1, S\} \quad \mathcal{G}_2 = \{\{S\} \cup C_{ad} \cup C_{bc}, V_T, P_2, S\}$$

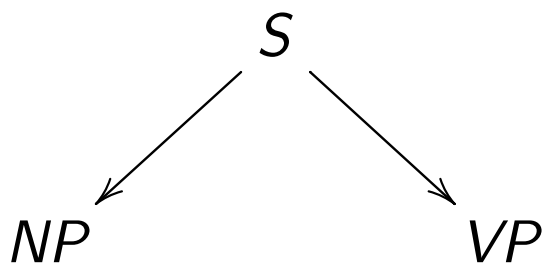
$\mathcal{G}_1$ generates all $a^n b^n c^m d^m$ with $n \neq m$ and some $a^n b^n c^n d^n$; $\mathcal{G}_2$ generates all $a^n b^m c^m d^n$ with $n \neq m$ and some $a^n b^n c^n d^n$

• then show (set theoretic argument) that both $\mathcal{G}_1$ and $\mathcal{G}_2$ must generate all but finitely many of the $a^n b^n c^n d^n$: all these have two different parse trees

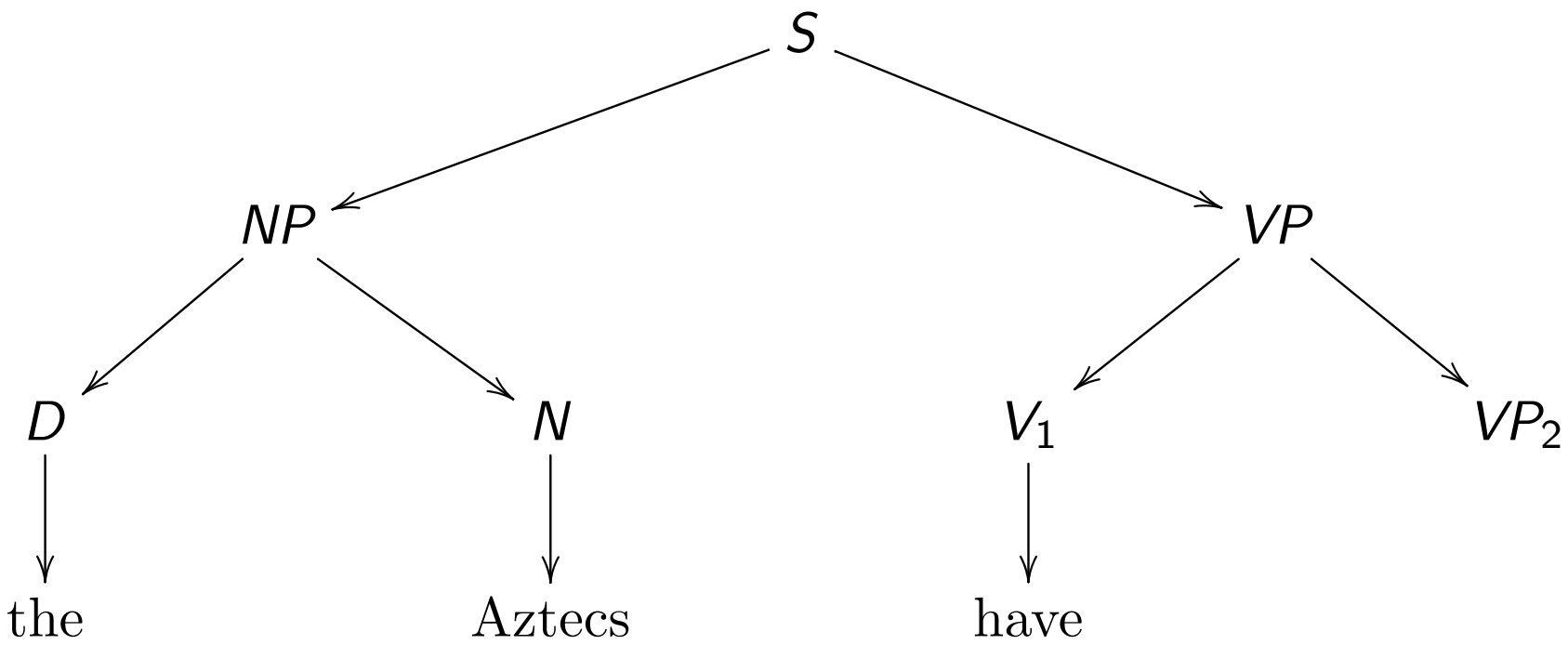
Parse trees and natural languages

Example How to *generate* the English sentence:
The book is believed to have been written by the Aztecs

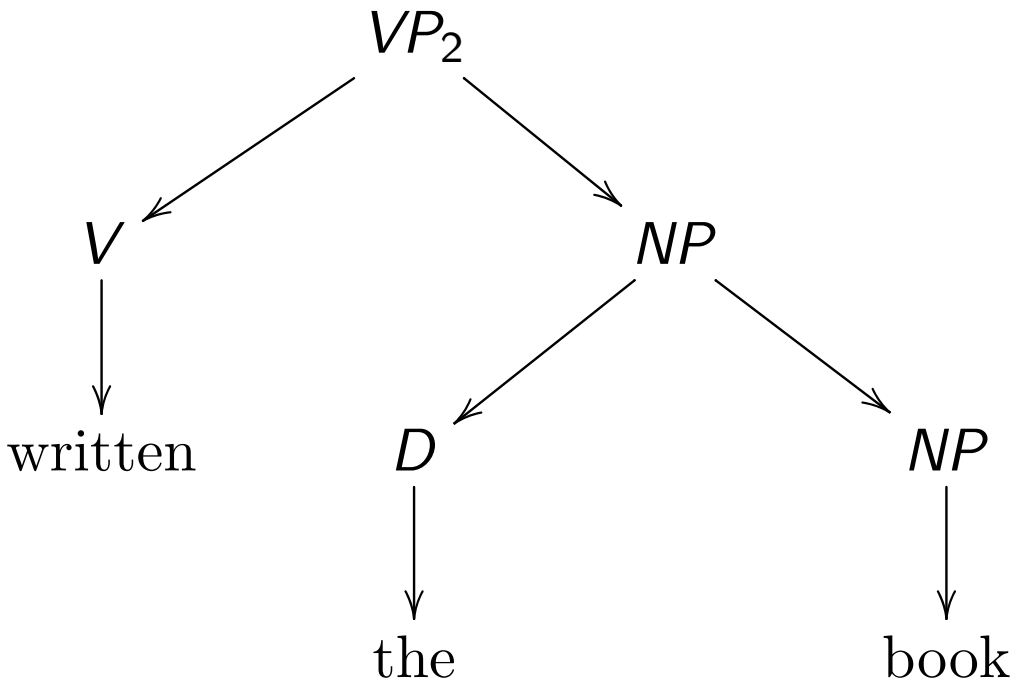
- Two step process:
  1. *generate* two separate sentences:
     (1) *The Aztecs have written the book*;
     (2) *We believe it*
  2. combine them with appropriate *transformations*
- first sentence (S): noun phrase (NP) + verb phrase (VP)



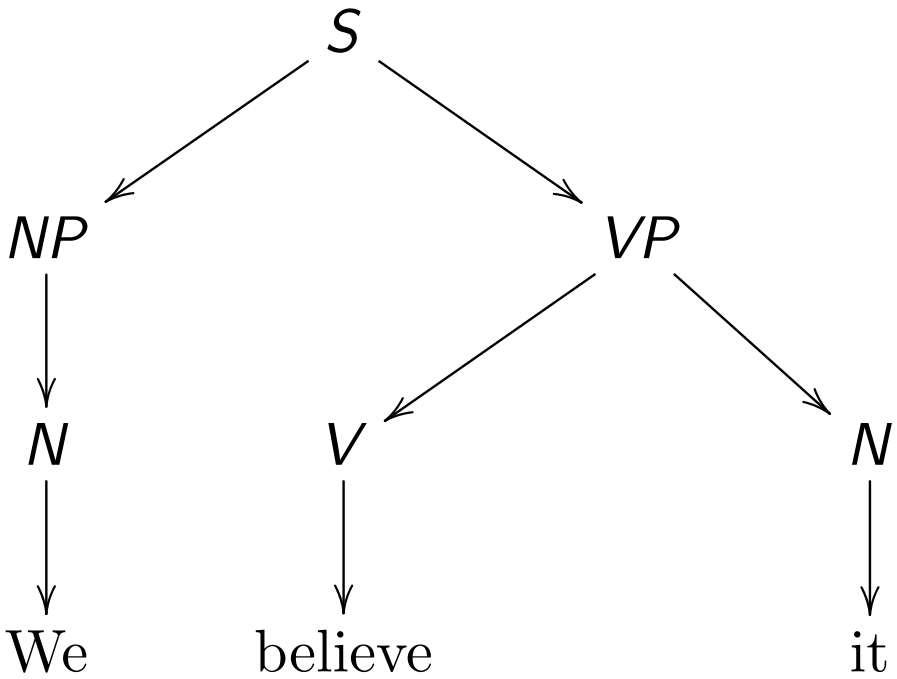
• The NP part is: determiner (D) + noun (N);
VP part has: auxiliary (V_1) + rest of phrase (VP_2)



- the VP_2 part consists of: verb (V) + noun phrase (NP)



- Similarly, the second sentence *We believe it* has a parse tree



- Operation 1: *Passive Transformation*

The Aztecs have written the book $\Rightarrow$ *The book has been written by the Aztecs*

- Operation 2: *Insertion*

We believe IT $\Rightarrow$ *We believe the book has been written by the Aztecs*

- Operation 3: *Passive Transformation*

We believe the book has been written by the Aztecs $\Rightarrow$ *The book is believed by us to have been written by the Aztecs*

- Operation 4: *Agent Deletion*

The book is believed by us to have been written by the Aztecs $\Rightarrow$ *The book is believed to have been written by the Aztecs*

Main idea:

Generative process with sentence (S) as start symbol; non-terminal symbols given by syntactic identifiers (NP, VP, N, V, D, etc.); terminals given by words; production rules encode syntactic structure, together with transformations on parse trees

Early formulation of Generative Grammar

1. Noam Chomsky, *The logical structure of linguistic theory* (1955), Plenum, 1975.
2. Noam Chomsky, *Syntactic structures*, Mouton, 1957.

Later developments focused more on transformations and less on production rules

A closer look at Transformational Grammar

• A set of trees (for example: parse trees of a context-free or context-sensitive grammar): *Base trees*

• finite, rooted, oriented, planar trees with decorated vertices: if one vertex v has only one outgoing edge e the label at $t(e)$ different from the label at $v = s(e)$

• Base trees $\mathcal{B} = \{B, V, V_T\}$ with B a collection of trees as above, $V_T \subset V$ a finite set of terminal symbols used to label leaves of trees in B and internal vertices labelled by non-terminal symbols $V_N = V \setminus V_T$

- Additional data $\Delta = \{\Sigma, \Sigma_A, X, V''\}$ with
  - Σ finite set of abstract symbols with $V_T \subseteq \Sigma$ and $\Sigma \cap V_N = \emptyset$
  - X abstract symbol not in $V \cup \Sigma$ (dummy variable)
  - a subset $\Sigma_A \subseteq \Sigma$
  - V'' set containing $V \cup \Sigma \cup \{X\}$ and additional symbols $Y^{(k)}$ with $Y \in V \cup \Sigma \cup \{X\}$ and $k \in \mathbb{N}$

Σ_A represents the set of symbols over which the language generated by the grammar is defined

- $\mathcal{R}$ = finite set of transformation rules (T-rules) (D, C) with respect to V'', Σ and X

T-rules

• symbols $X, X^{(k)}$ in V'' mark parts of the tree that cannot be moved by the transformation T

• $D =$ domain statement: string $\alpha_1 \cdots \alpha_k$ of symbols in V''

• $C =$ structural change statement on D: string $\beta_1 \cdots \beta_k$ of symbols in $\{k\}_{k \in \mathbb{N}} \cup \Sigma$

$\beta_j = j$ if symbol $D_j = \alpha_j$ of D is some $X^{(r)}$ (unmoved by T)
otherwise β_j is either some $i \neq j$ or some symbol in Σ

- Example: passivization in English

the cat ate the mouse $\mapsto$ *the mouse was eaten by the cat*

$N^{(1)}\,TVN^{(2)} \mapsto N^{(2)}\ T$ be $E_n\ VN^{(1)}$

$N^{(1)} =$ cat, $T =$ tense, past; $V =$ eat, $N^{(2)} =$ mouse

$TV \mapsto T$ be $E_n V$ ate $\mapsto$ was eaten

- T_{pass} rule (D, C) where

$D = \alpha_1\alpha_2\cdots\alpha_8 = X^{(1)}\$N^{(1)}\,TVN^{(2)}\$X^{(2)}$

$C = \beta_1\beta_2\cdots\beta_8 = 1264(\ \text{be}\ E_n\ 5\ \text{by}\ )378$

$\$ =$ boundary marker

States

• additional structure of transformational grammar:
$\Omega = \{K, \mathcal{N}, \delta, s_0\}$

- K = finite set of states, s_0 = start state
- $\mathcal{N} = \{N(s), s \in K\}$ with $N(s)$ partially ordered set over $\mathcal{R} \cup \{\#\}$ (with $\#$ stop symbol occurring as maximal element)
- $\delta : K \times \mathcal{R} \to K$ (next state function)

Keeps into account order of application of the T-rules (order matters)

Records the "past history" of the use of the rules (can reconstruct the path of rule applications)

Assume a rule T leaves a tree unchanged if it does not apply to it (continue to next rule in the ordered list)

Language generated by a transformational grammar

- the set of base trees = *deep structure*
- all the tree produced by applying compositions of transformations to base trees = *surface structure*

- (τ, s) with τ a tree and $s \in K$

$$(\tau, s) \vdash (\tau', s')$$

if $\exists\ T = (D, C)$ T-rule with $\tau' = T(\tau)$, $T \in N(s)$, $s' = \delta(s, T)$, there is no other τ'' and $T' \in N(s)$ with $T' < T$ and $\tau'' = T'(\tau)$

- string w generated by T-grammar if $w \in \Sigma_A^*$, there are τ, τ' and s' with $\tau \in \mathcal{B}$, $(\tau, s_0) \vdash^* (\tau', s') \vdash S\mathit{top}$ and w is the terminal string of the tree τ'

Example of cross-serial dependencies

dat Jan Piet de kinderen zag helpen zwemmen

(Jan saw Piet help the children swim)

- core construction using CFG (context-free grammar)

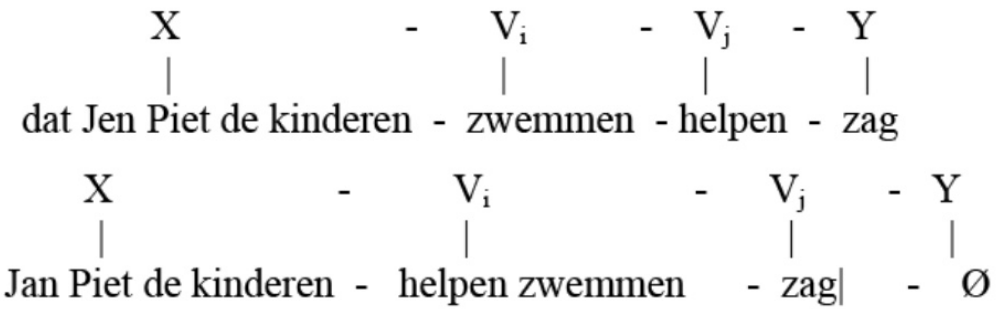
$$S \to NP\ VP\,; \quad VP \to S\ V\,; \quad VP \to V$$

- these alone give ungrammatical strings

dat [$_S$ Jan [$_{VP}$ [$_S$ Piet [$_{VP}$[$_S$ de kinderen [$_{VP}$ zwemmen]] helpen]] zag]]
that Jan Piet the children swim help saw

- then transformational rules (Verb Raising)

$$X - V_1 - V_2 - Y \to X - [_V\, V_2 - V_1] - Y$$



Refences

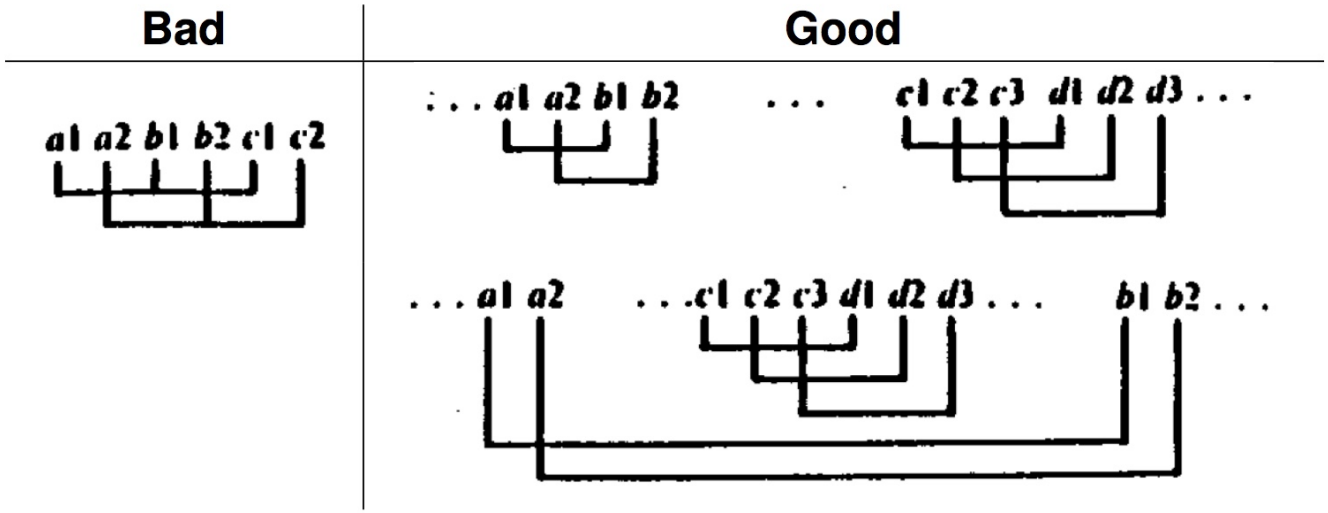
- Noam Chomsky, *Selected Readings on Transformational Theory*, Dover 2012.
- Seymour Ginsburg, Barbara Partee, *A mathematical model of Transformational Grammars*, Information and Control 15 (1969) 297–334
- P.S.Peters, R.W.Ritchie, *On the generative power of transformational grammars*, Information Sci. 6 (1973), 49–83.
- Barbara Partee, Alice ter Meulen, Robert Wall, *Mathematical Methods in Linguistics*, Kluwer, 1990.
- last example from
  Knut Tarald Taraldsen, *Generative Grammar*, Oxford Research Encyclopedias, Linguistics, 2016

Computational Minimalism and Merge Grammars
Some References:

- E.P. Stabler, *Computational perspectives on minimalism*, in "Oxford Handbook of Linguistic Minimalism", Oxford University Press, 2010, 616–641.
- K. Vijay-Shanker, D. Weir, *The equivalence of four extensions of context free grammar formalisms*, Mathematical Systems Theory, 27 (1994) 511–545.
- P. beim Graben, S. Gerth, *Geometric representations for minimalist grammars*, arXiv:1101.5076
- T. Hunter, C. Dyer, *Distributions on Minimalist Grammar Derivations*, Proc. 13th Meeting of the Mathematics of Language (MoL 13), Association for Computational Linguistics, 2013, pp.1–11.
- S. Indurkhya, *Incremental Acquisition of a Minimalist Grammar using an SMT-Solver*, 2022.

Extend the Context-Free Class to Mild Context Sensitivity

- limited cross-serial dependencies



- polynomial time parsing
- semilinearity

Semilinearity

• a subset $\mathcal{V} \subset \mathbb{Z}_+^k$ is semilinear if it is a finite union of sets of the form

$$\{c + \sum_{w \in \mathcal{P}} \lambda_w \, w \mid c \in \mathcal{C}\}$$

for some finite sets $\mathcal{C}, \mathcal{P} \subset \mathbb{N}^k$ and scalars λ_w

• a language $\mathcal{L} \subset \mathfrak{A}^*$ with alphabet $\#\mathfrak{A} = k$ is semilinear iff for any monoid homomorphism

$$\varphi : (\mathfrak{A}^*, \star) \to (\mathbb{Z}_+^k, +)$$

the image $\varphi(\mathcal{L}) \subset \mathbb{N}^k$ is a semilinear set

• context-free and tree-adjoining grammars have semilinear property (Joshi and Yokomori, 1983)

Multiple Context Free Grammars (MCFG)

- introduced by H.Seki, T.Matsumura, M.Fujii, T.Kasami, 1990
- Example: $\mathcal{L} = \{a_1^n a_2^n \cdots a_{2m}^n \mid n \geq 0\}$ is an m-MCFG

$$\mathcal{G} = (N = \{A, S\}, T = \{a_i\}_{i=1}^{2m}, O = \cup_{k=1}^{m}(T^*)^k, \{f, g\}, P, S)$$

with production rules P

$$f(x_1, x_2, \ldots, x_m) = (a_1 x_1 a_2, a_3 x_2 a_4, \ldots, a_{2m-1} x_m a_{2m})$$

$$g(x_1, x_2, \ldots, x_m) = x_1 x_2 \cdots x_m$$

$$A \to (\epsilon, \epsilon, \ldots, \epsilon), \quad A \to f[A], \quad S \to g[A]$$

MCFG: general definition $\mathcal{G} = (N, T, O, F, P, S)$

- $O = \cup_{k=1}^{m} (T^*)^k$
- finite set F of (partial) functions $f : O^{a(f)} \to O$ some $a(f) \in \mathbb{N}$
- $f \in F$ function of $a(f)$ variables: there are $0 \leq r(f), d_k(f) \leq m,\ k = 1, \ldots, m,$

$$f : \prod_{k=1}^{m} (T^*)^{d_k(f)} \to (T^*)^{r(f)}$$

- functions $f(x_1, \ldots, x_{a(f)})$ are concatenations of constant strings in T^* and variables in $X = \{x_{kj}, k = 1, \ldots, a(f), j = 1, \ldots, d_k(f)\}$ with each x_{ij} occurring at most once
- $d : N \to \mathbb{N}$, $d(S) = 1$, if $A \to f(A_1, \ldots, A_{a(f)})$ in P then $r(f) = d(A)$ and $d_k(f) = d(A_k)$

Properties:

- m-MCFG $\subsetneq$ $(m+1)$-MCFG
- MCFGs are semilinear (Vijay-Shanker, Weir, Joshi, 1987)
- tree adjoining grammars sit between CFG and 2-MCFG

$$\text{CFG} = \text{1-MCFG} \subsetneq \text{TAG} \subsetneq \text{2-MCFG}$$

- recognition $w \in \mathcal{L}_{\mathcal{G}}$ is polynomially decidable
  (but inclusion $\mathcal{L}_{\mathcal{G}_1} \subseteq \mathcal{L}_{\mathcal{G}_2}$ is undecidable)
- MCFGs can be made stochastic as CFGs

Merge Grammars or Minimalist Grammars (MG)

- introduced in
  - Edward P. Stabler and Edward L. Keenan, *Structural similarity within and among languages*, Theoretical Computer Science, 293 (2003) 345–363.
- formalizing the derivations within Chomsky's Minimalist Model in the setting of formal languages
- proved to be equivalent in terms of generative capacity to mCFG but not equivalent in terms of computational complexity: Merge description is exponentially more succinct than mCFGs
  - R.C.Berwick, *Mind the Gap*, 2015.

- Minimalist Grammar $\mathcal{G} = (\mathfrak{A}, Sel, Lic, Lex, c)$
  - $\mathfrak{A}$ finite alphabet
  - *Lic* (licensing types) and *Sel* (selecting types) disjoint finite sets
  - Syn set of syntactic features:

$$
\begin{aligned}
selectors &= \{=f \mid f \in Sel\} \\
selectees &= \{\ \ f \mid f \in Sel\} \\
licensors &= \{+f \mid f \in Lic\} \\
licensees &= \{-f \mid f \in Lic\}
\end{aligned}
$$

  - lexicon finite subset
    $Lex \subset \mathfrak{A}^* \times (\text{selectors} \cup \text{licensors})^* \times \text{selectees} \times \text{licensees}^*$
  - $c \in Sel$ type for completed expression

Examples of minimalist lexicon items in *Lex*

pierre : d	*who* : d −wh
marie : d	*will* : =v =d t
praise : =d v	ϵ : =t c
often : =v v	ϵ : =t +wh c

- lexical categories: adjective A, adjective phrase AP, adverb Adv, adverb phrase AdvP, noun N, noun phrase NP, verb V, verb phrase VP, etc.
- functional categories: coordinate conjunction C, determiner D, negation Neg, particle Par, preposition P, prepositional phrase PP, subordinate conjunction Sub, tense T, tense phrase TP, etc.
- selection: =X selection of an X phrase
- licensees: -X requirements forcing movement
- licensors: features that satisfy licensees requirements like +wh +case etc.

Operations instead of production rules only two fixed kinds of operations in Minimalist Grammars

1. MERGE
2. MOVE

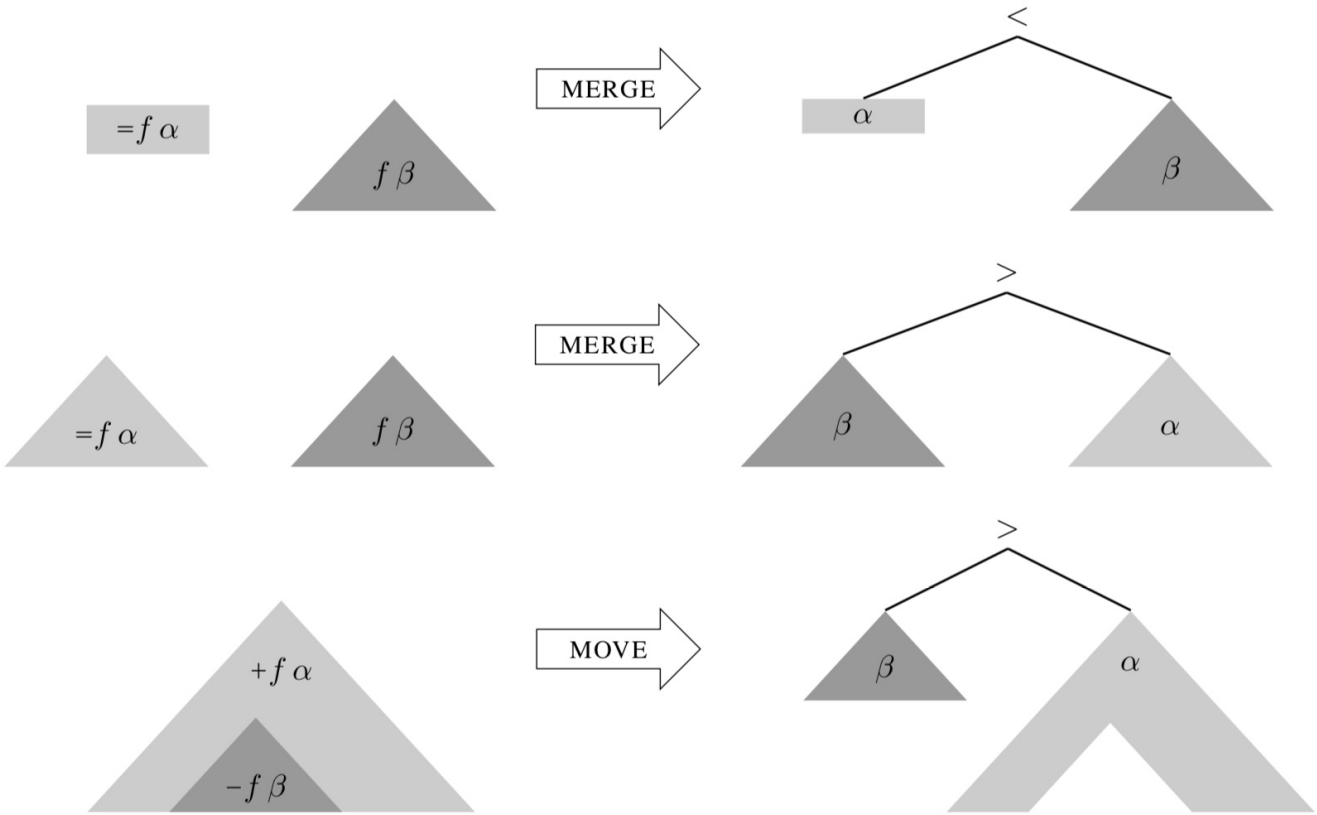


Figure 2: Graphical illustrations of definitions of MERGE and MOVE. Rectangles represent single-node trees. Triangles represent either single-node trees or complex trees, but the second case of MERGE applies only when the first case does not (i.e. when the $=f\ \alpha$ tree is complex).

Merge and Move

- more formal description of MERGE and MOVE

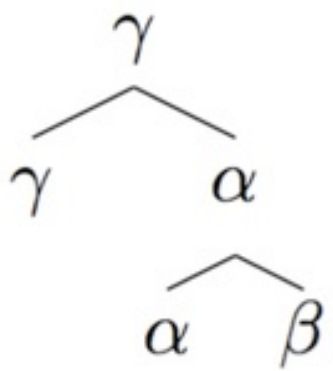
$$\textsc{merge}\big(e_1[{=}f\ \alpha], e_2[f\ \beta]\big) = \begin{cases} [_< e_1[\alpha]\ e_2[\beta]] & \text{if } e_1[{=}f\ \alpha] \in Lex \\ [_> e_2[\beta]\ e_1[\alpha]] & \text{otherwise} \end{cases}$$

$$\textsc{move}\big(e_1[{+}f\ \alpha]\big) = [_> e_2[\beta]\ e_1'[\alpha]]$$

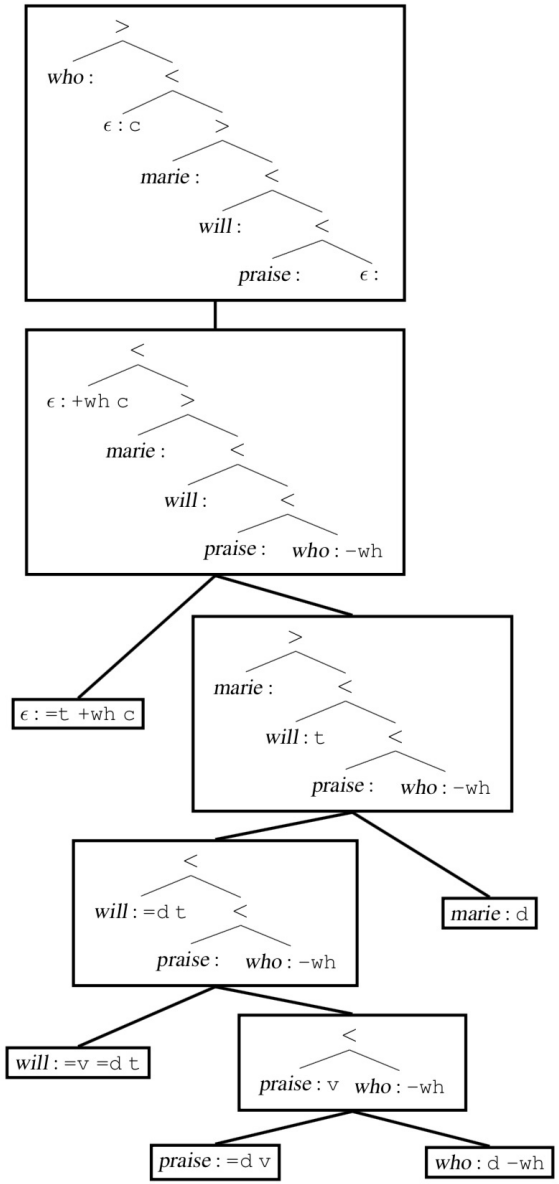
where $e_2[{-}f\ \beta]$ is a unique subtree of $e_1[{+}f\ \alpha]$

and e_1' is like e_1 but with $e_2[{-}f\ \beta]$ replaced by an empty leaf node $\epsilon : \epsilon$

- Merge: $(\alpha, \beta) \mapsto \{\alpha, \{\alpha, \beta\}\}$ or $\{\beta, \{\alpha, \beta\}\}$
- iterations: $(\gamma, \{\alpha, \{\alpha, \beta\}\}) \mapsto \{\gamma, \{\gamma, \{\alpha, \{\alpha, \beta\}\}\}\}$

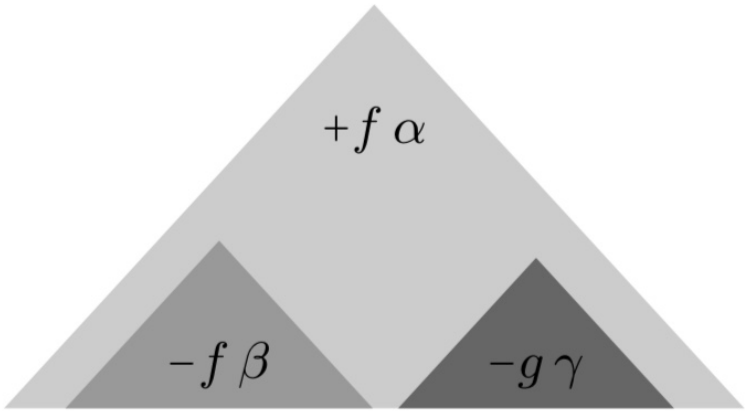


Example of derivation in Minimalist Grammars (embedded question)



MG = MCFG (Theorem 1 of Stabler 2010)

Main Idea of how to transform a MG grammar into a MCFG

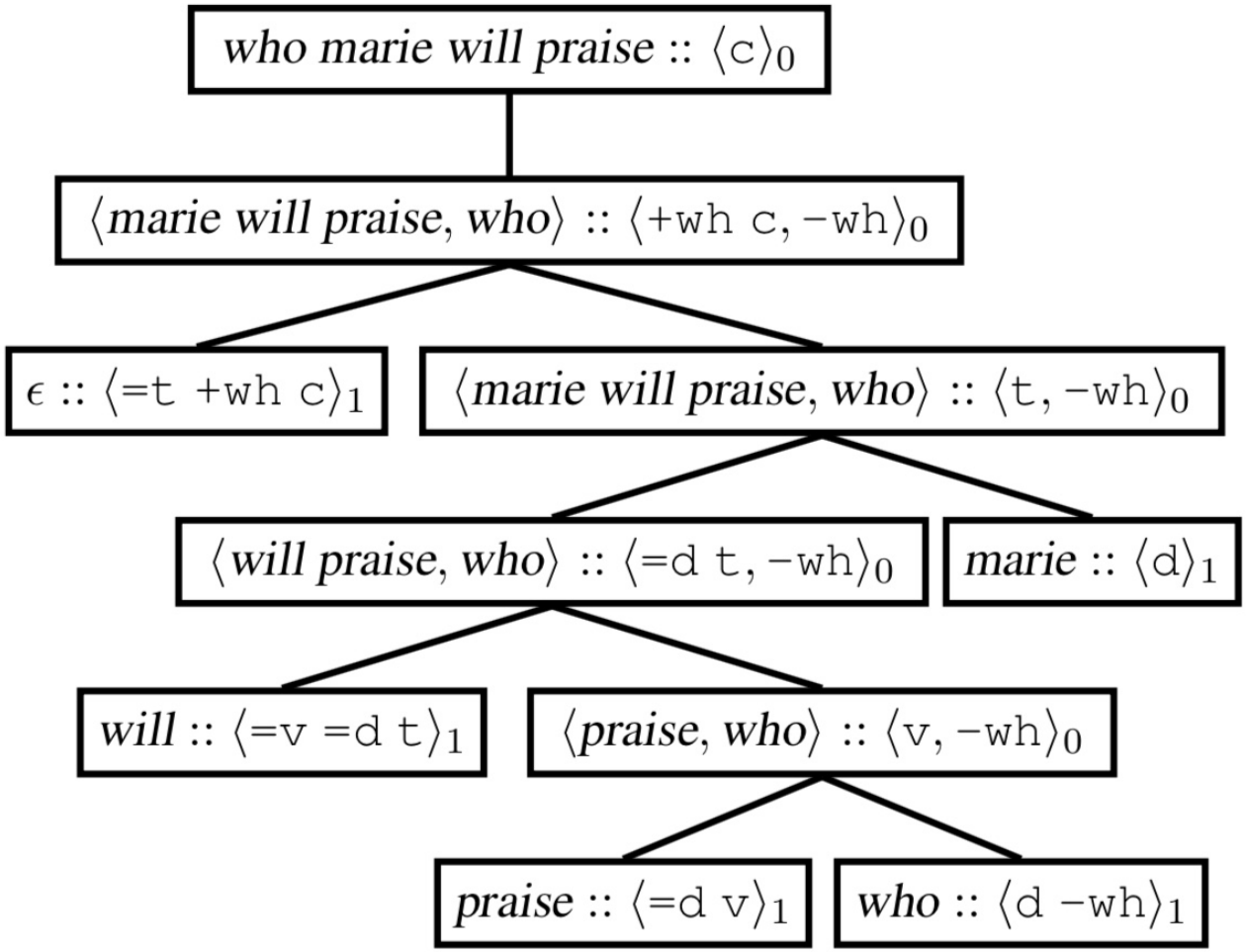


$$\langle \quad s :: +f\ \alpha \quad , \quad t : -f\ \beta \quad , \quad u : -g\ \gamma \quad \rangle_0$$

transform trees into tuples of strings (subscript 0: non-lexical expressions; subscript 1 lexical; also :: and : for lexical and derived)

- these tuples of strings give the production rules of a MCFG
- start symbol of MCFG is $\langle c \rangle_0$

Example: previous derivation in terms of tuples of strings



External and Internal Merge Operations

• MG operations of MERGE and MOVE unified as two aspects of the same *merge* operation

1. *external merge*

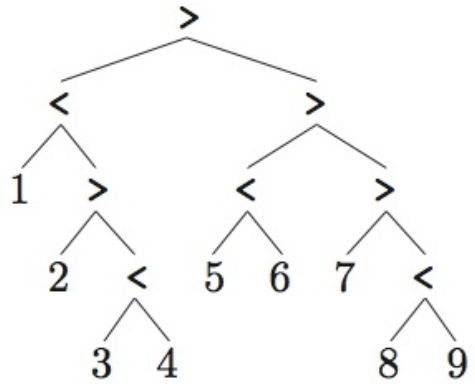
$$\mathbf{em}(t_1[\texttt{=x}], t_2[\texttt{x}]) = \begin{cases} \overset{<}{\wedge}\ t_1 \quad t_2 & \text{if } |t_1| = 1 \\ \overset{>}{\wedge}\ t_2 \quad t_1 & \text{otherwise} \end{cases}$$

2. *internal merge*

$$\mathbf{im}(t_1[\texttt{+x}]) = \overset{>}{\wedge}\ t_2^M \quad t_1\{t_2[\texttt{-x}]^M \mapsto \epsilon\}$$

under *shortest move constraint* (SMC): exactly one head in the tree has $-x$ as first feature; t^M maximal projection

Head and Projection



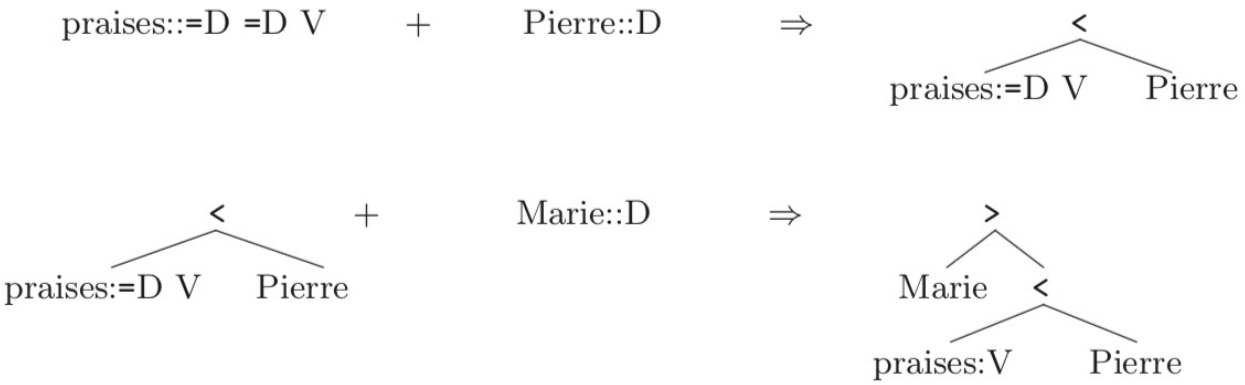
- labels $>$ and $<$ of merge identify where *head* of the tree is: here leaf vertex number 8
- *maximal projection* in T is a subtree of T that is not a proper subtree of any larger subtree with the same head
- leaves $\{2, 3, 4\}$ determine a subtree with head vertex the leaf numbered 3
- any larger subtree in T would have a different head: this is a maximal projection
- also subtree determined by leaves $\{5, 6\}$

Notation about features

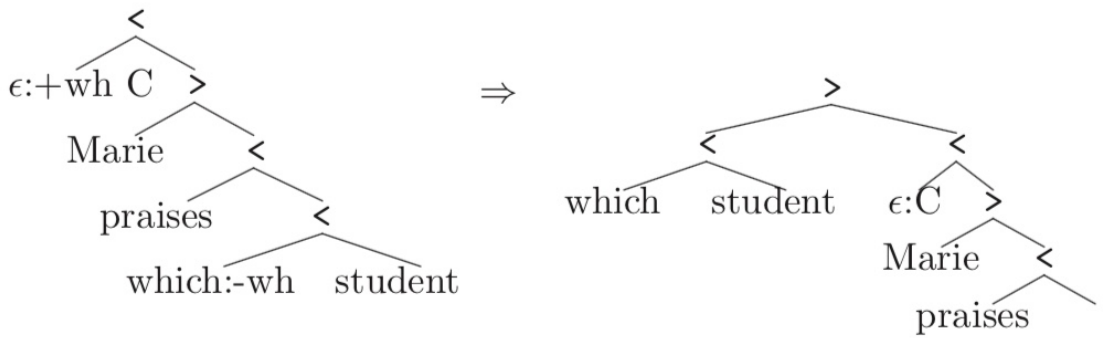
- in addition to labels $\{>, <\}$ of merge operations, finite set of syntactic features labels $X \in \{N, V, A, P, C, T, D, \ldots\}$
- *selector* features denoted by the symbol σX for a head selecting a phrase XP
  (Note: usually notation $= X$ rather than σX for selector)
- $T[\alpha]$ tree where head is labelled by an ordered set of syntactic features starting with α
- operation $T[\alpha] \mapsto T$ removes the α-feature from the head vertex

Examples

- external merge



- internal merge



Merge and the Origin of Language

- Robert C. Berwick, Noam Chomsky, *Why only us?* MIT Press, 2015.

• proposal of a single significant evolutionary change leading to the structure of human languages in a single computational operation capable of generating recursive structures: *merge operation*

• is there a way to characterize merge as a fundamental structure of recursion in a mathematical sense?

Note: we will return to this fundamental question in presenting the current theory of Merge

Coming up next: a digression on probability

- use of probabilities in formal languages
- more general aspects of probability and information in (computational) linguistics
- probability in the study of deterministic systems