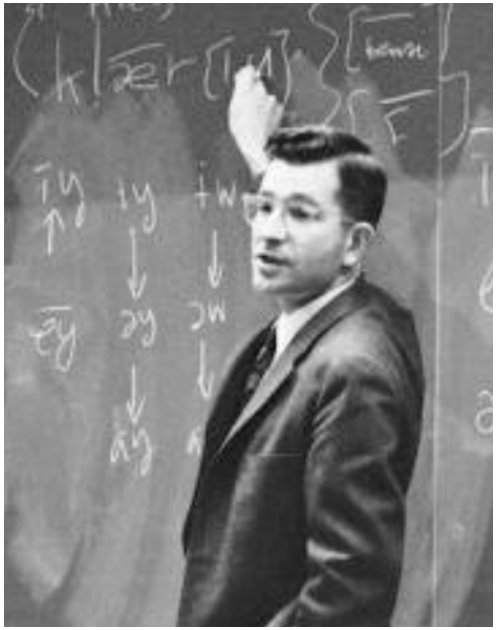
Lecture 2: Generative Linguistics
Ma 191c: Mathematical Models of Generative Linguistics

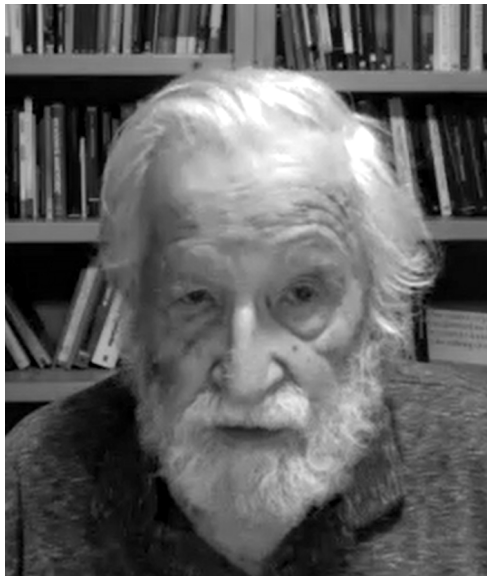
Matilde Marcolli

Caltech, Spring 2024

Generative Linguistics – The History

the field was initiated and developed by **Noam Chomsky**





Formal languages, transformational grammar (1950s-1970s):

- N. Chomsky, *The Logical Structure of Linguistic Theory*, 1955.
- N. Chomsky, *Syntactic structures*, Mouton 1957
- N. Chomsky, *Aspects of the theory of syntax*, MIT Press, 1965

Principles and parameters, Government and Binding (1980s):

- N. Chomsky, *Lectures on Government and Binding*, Mouton de Gruyter, 1981

Minimalism (1990s):

- N. Chomsky, *The Minimalist Program*, MIT Press, 1995.

New Minimalism: Merge and SMT (starting ∼2000, then 2013-now):

- N. Chomsky et al., *Merge and the Strong Minimalist Thesis*, Cambridge Elements, 2023.

Some general aspects

language (in human brains) is a complex and highly structured phenomenon: modeling production, acquisition, parsing

- main focus on *syntax*
  (though generative models of phonology, morphology/morphosyntax, semantics have been studied)
- what is the fundamental *generative process* of language? (generative grammar)
- language acquisition and "parameter setting"
- syntax as structure

What does it mean to model? what does it mean to explain? fundamental question about the nature and purpose of science

The Goal: understanding language as a highly structured and complex computational generative process

Why mathematics?

- mathematics is the study of *structures*
- but isn't it about *numbers*? no... numbers matter in mathematics only because they have interesting structures
- ... *and so does language*

most important research problems in mathematics revolve around understanding different kinds of structures (numbers, geometric objects, abstract algebraic properties)

structures understood by mathematics played a fundamental role in the development of *theoretical physics* in the 20th century...

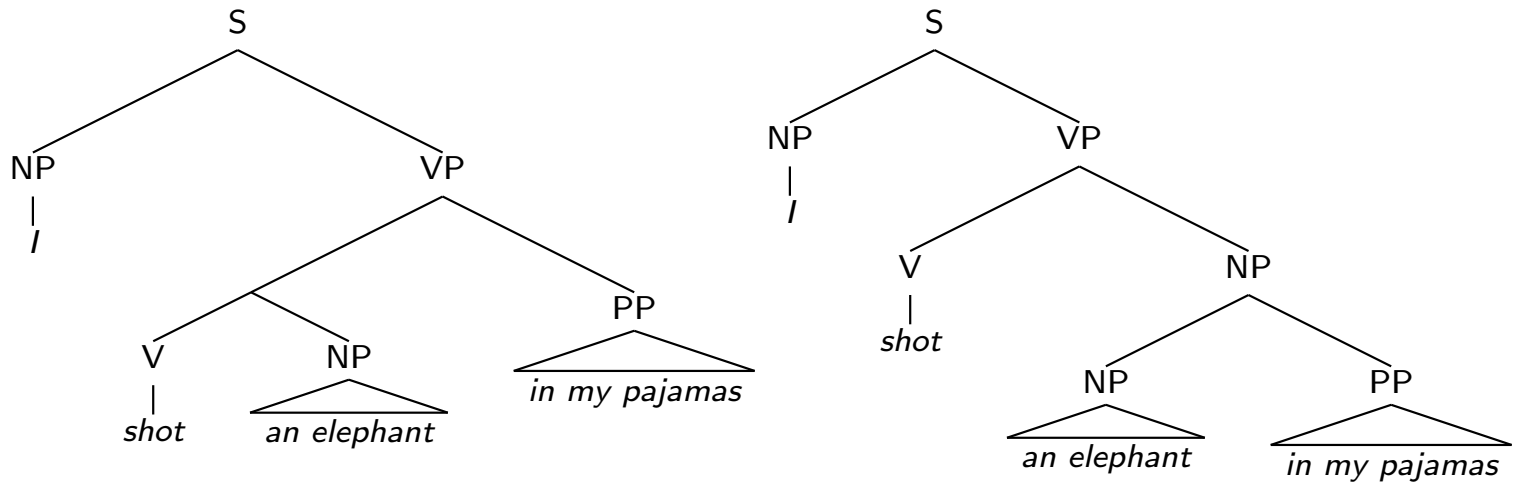
can linguistics be studied with the same principles and methods as theoretical physics? (see N.Chomsky "Rules and Representations", 1980)

How is syntax "a structure"?

Example: look at this sentence

I shot an elephant in my pajamas – what it was doing in my pajamas, I'll never know (Groucho Marx)

... why is it funny? because it conflates two different structures



a sentence is **not** just a string of words!

we perceive *structural relations* not *proximity relations* in the ordering of words: structure reflects the different possible way sentences are *generated*

Generative Linguistics

... Chomsky describes it as the study of
language as a structure of discrete infinity

Why discrete (countable) infinity?

- Sentence formation in language allows for recursions
- for example, you can "help someone" and you can "help someone help someone" and you can "help someone help someone help someone..."
- in practice we would not *use* arbitrarily long iterations **but** our brain *immediately understands* that such a recursion
  1. could continue
  2. would remain grammatically correct
  3. would remain meaningful
- recursions are the telltale sign of a computational process

Early developments (1950s-1960s): Formal Lanaguages

- formal languages describe *strings of words* as produced by a computational mechanism (automaton)
- different classes of formal languages can be computed by different classes of automata (Chomsky hierarchy)
- *different kinds of possible recursions* in different classes
- programming languages, presentations of discrete groups, etc also give rise to formal languages
- which classes of formal languages can describe natural (human) languages?

Formal Languages – Grammar: a quadruple $\mathcal{G} = (V_N, V_T, P, S)$

- 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*

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

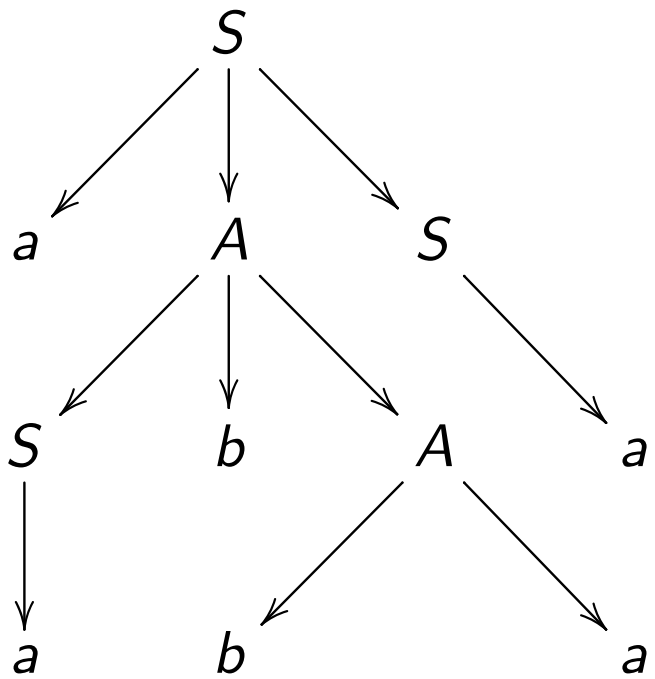
$$\mathcal{L}_{\mathcal{G}} = \{w \in V_T^\star \mid S \overset{\bullet}{\to}_P w\}$$

language with alphabet V_T

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 derivation in $\mathcal{G}$ for the string $aabbaa$ in $\mathcal{L}_{\mathcal{G}}$



The Chomsky hierarchy

Types:

- Type 0: *unrestricted grammars*
- Type 1: *context-sensitive grammars*
- Type 2: *context-free grammars*
- Type 3: *regular grammars*

Machine Recognition:

- Type 0: Turing machine
- Type 1: linear bounded automaton
- Type 2: non-deterministic pushdown stack automaton
- Type 3: finite state automaton

N.Chomsky, "Formal properties of grammars", 1963

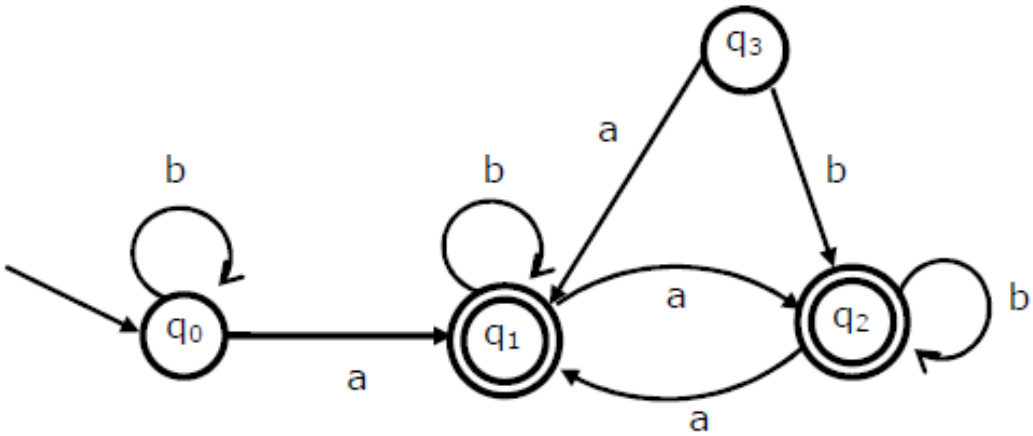
Context free and context sensitive production rules

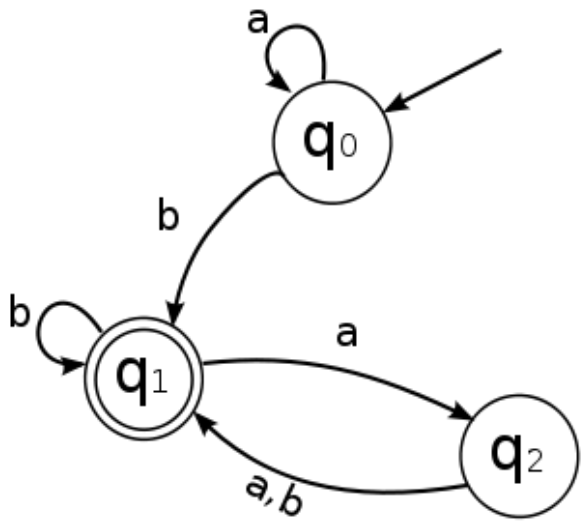
- context free: $A \to \alpha$ with $A \in V_N$ and $\alpha \in (V_N \cup V_T)^\star$
- context sensitive: $\beta A \gamma \to \beta \alpha \gamma$ with $A \in V_N$
$\alpha, \beta, \gamma \in (V_N \cup V_T)^\star$ and $\alpha \neq \epsilon$

context free is context sensitive with $\beta = \gamma = \epsilon$

"context free" languages: a first attempt to model natural languages (Chomsky, 1956); some programming languages in this class (e.g. Fortran, Algol, HTML)

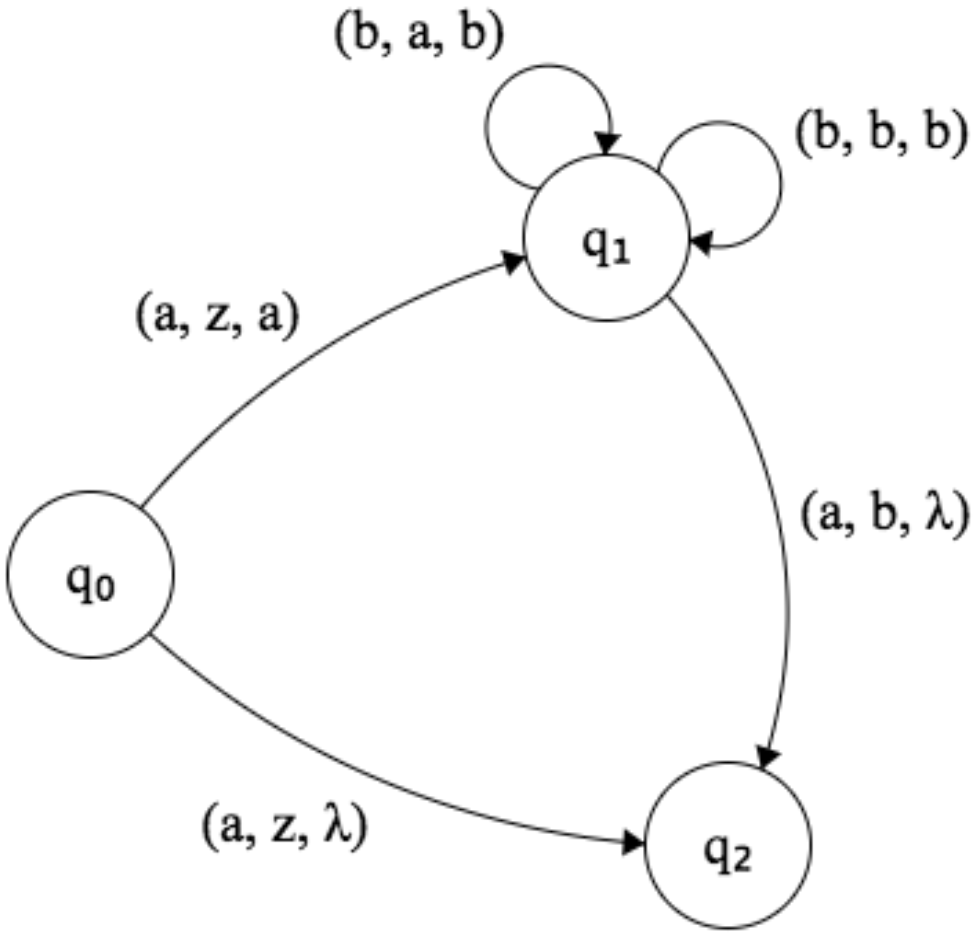
Example: finite state automata





with marked final states

Example: pushdown stack automaton



transition (a, b, c): read letter b on top of memory stack, remove b and place c at the top of the stack: move from $(q_i, aw, b\alpha)$ to $(q_j, w, c\alpha)$

Examples of recursions in different classes

• Type 3 (regular): $\mathcal{G} = (\{S, A\}, \{0, 1\}, P, S)$ with productions P given by

$$S \to 0S, \quad S \to A, \quad A \to 1A, \quad A \to 1$$

then $\mathcal{L}_{\mathcal{G}} = \{0^m 1^n \mid m \geq 0, n \geq 1\}$

• Type 2 (context-free): $\mathcal{G} = (\{S\}, \{0, 1\}, P, S)$ with productions P given by

$$S \to 0S1, \quad S \to 01$$

then $\mathcal{L}_{\mathcal{G}} = \{0^n 1^n \mid n \geq 1\}$

• Type 1 (context-sensitive): $\mathcal{G} = (\{S, B, C\}\{a, b, c\}, P, S)$ with productions P

$$S \to aSBC, \quad S \to aBC, \quad CB \to BC,$$

$$aB \to ab, \quad bB \to bb, \quad bC \to bc, \quad cC \to cc$$

the $\mathcal{L}_{\mathcal{G}} = \{a^n b^n c^n \mid n \geq 1\}$

Main Idea: a generative grammar $\mathcal{G}$ determines *what kinds of recursive structures* are possible in the language $\mathcal{L}_{\mathcal{G}}$

Representing natural languages?

• Question: How good are context-free grammars at representing natural languages?

- Originally conjectured to be the right class for natural languages
- Not always good, but often good (better than earlier criticism indicated)
- Some explicit examples not context-free:

1. Riny Huijbregts (first counterexample: Dutch is context sensitive)
2. S. Shieber (also Swiss German is context sensitive)

cross-serial subordinate clause

Some natural languages are context-sensitive

• Try to show not context-free by finding cross-serial dependencies of arbitrarily large size (realizable by context-sensitive not by context-free)

• Example: the language $\mathcal{L} = \{xx^R \mid x \in \{a, b\}^*\}$ has cross serial dependencies of arbitrary length (the i-th and $(n + i)$-th term have to be the same (x^R = reversal of x)

• if cross serial dependencies of arbitrary length not context-free

Swiss German cross-serial order in dependent clauses

$$wa^n b^m x c^n d^m y$$

Jan säit das mer (d'chind)^n (em Hans)^m es huus haend wele (laa)^n (häfte)^m aastrüche
non-context-free language

same type of example works for Dutch

- Context-free class too small
- Context-sensitive class too large
- Intermediate candidates: tree-adjoining grammars, multiple context-free grammar, etc

Other Problem: Clearly there are many more formal languages that do not correspond to natural (human) languages (even within the appropriate class that contains natural languages)

Example: Programming Languages: Fortran is context-free; C is context-sensitite; C^{++} is Type 0, ...

Examples: Formal Languages constructed from finitely presented discrete groups

Formal Language of a finitely presented group

- Group G, with presentation $G = \langle X \mid R \rangle$ (finitely presented)
  - X (finite) set of generators $x_1, \ldots, x_N$
  - R (finite) set of relations: $r \in R$ words in the generators and their inverses
- for $G = \langle X \mid R \rangle$ call $\hat{X} = \{x, x^{-1} \mid x \in X\}$ symmetric set of generators
- Language associated to a finitely presented group $G = \langle X \mid R \rangle$

$$\mathcal{L}_G = \{w \in \hat{X}^\star \mid w = 1 \in G\}$$

set of words in the generators representing trivial element of G

- Question: What kind of formal language is it?

• Algebraic properties of the group G correspond to properties of the formal language $\mathcal{L}_G$:

1. $\mathcal{L}_G$ is a regular language (Type 3) iff G is finite (Anisimov)
2. $\mathcal{L}_G$ is context-free (Type 2) iff G has a free subgroup of finite index (Muller–Schupp)

Example: Take $G = \mathrm{SL}_2(\mathbb{Z})$, infinite so $\mathcal{L}_G$ not regular; generators

$$S = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \text{ and } T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

with relations S^2 and $(ST)^3$

$$\begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix} \text{ and } \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}$$

generate a free subgroup F_2 of index 12 in $\mathrm{SL}_2(\mathbb{Z})$ (of index 2 in $\Gamma(2)$ that has index 6 in $\mathrm{SL}_2(\mathbb{Z})$) so $\mathcal{L}_{\mathrm{SL}_2(\mathbb{Z})}$ is context-free

The "Boundaries of Babel" Problem

- Given a class of formal languages good enough to contain natural languages
- How to characterize the "region" within this class of formal languages that is populated by actual human (natural) languages?
- What is the *geometry* of the space of natural languages inside the space of formal languages?

• Andrea Moro, *The Boundaries of Babel. The Brain and the Enigma of Impossible Languages*, Second Edition, MIT Press, 2015

Want: a characterization and parameterization of the syntax of human languages

- formal languages are now no longer viewed as a good way to model generative syntax
  1. focused on strings rather than structures
  2. modeling parts of syntax require complicated sets of production rules
  3. too many formal languages in each class not corresponding to natural languages
  4. a simpler and more conceptual model is preferred
- still very useful in other parts of linguistics (eg phonology)
- still useful to model and distinguish recursions
- can be useful for modeling constraints
- versions including tree structures are also used (also graph grammars)
- used in computer science

More details on the mathematics of formal languages in next lectures

Sketch of Generative Syntax beyond formal languages

- Transformational grammar
- Principles and Parameters
- Government and Binding
- Minimalist Program
- Computational Minimalism
- Merge and the Strong Minimalist Thesis

The last topic will be the main focus of this class

These previous historical steps: quick review here and in the next preliminary lectures

Transformational Grammar (Chomsky, 1957)

- Sentences have two levels of structure: *deep structure* and *surface structure*
- *deep structure*: closer to semantic level, properties common across languages, mapped to surface structure via *transformations* that operate on parse trees underlying sentences
- *surface structure*: language specific
- a transformational grammar is a system of tree automata
- in more recent theories (minimalist program), deep structure and surface structure replaced by *logical form* and *phonetic form*

...more details later, after discussing formal languages

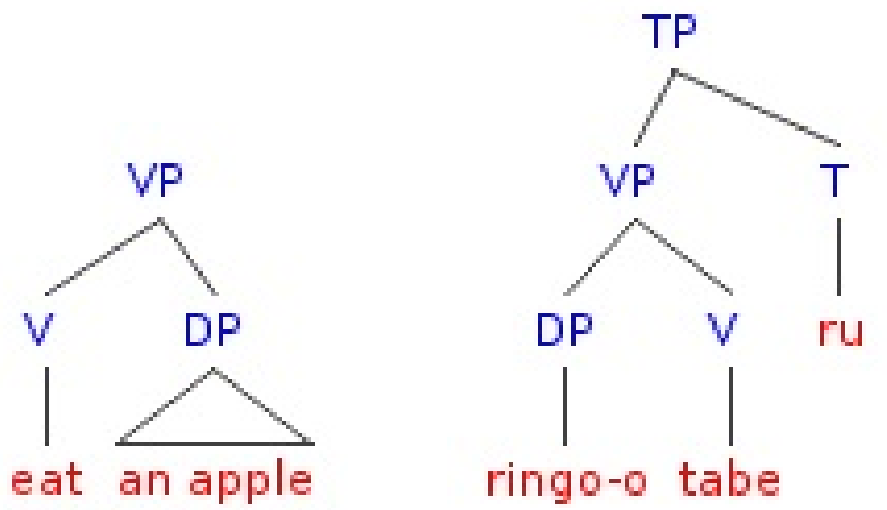
Government and Binding (Principles and Parameters)

(Chomsky, 1981)

- *principles*: general rules of grammar
- *parameters*: binary variables (on/off switches) that distinguish languages in terms of syntactic structures
- Example of parameter: head-directionality

(head-initial versus head-final)

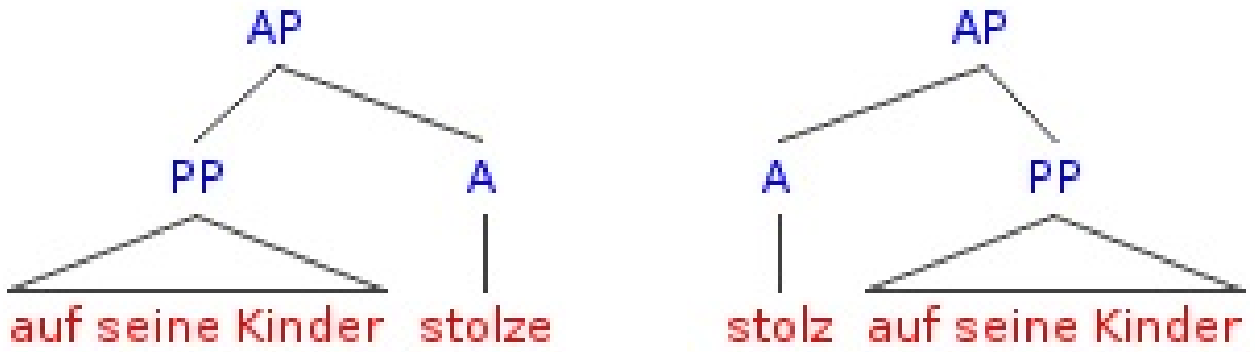
English is head-initial, Japanese is head-final



VP= verb phrase, TP= tense phrase, DP= determiner phrase

...but not always so clear-cut: German can use both structures

auf seine Kinder stolze Vater (head-final) or
er ist stolz auf seine Kinder (head-initial)



AP= adjective phrase, PP= prepositional phrase

- effect of interaction with other syntactic parameters (V2)
- Corpora based statistical analysis of head-directionality (Haitao Liu, 2010) in various languages shows a continuum between head-initial and head-final

Examples of Principles

- *Structure Preservation Principle*: identifies transformations preserving deep structure (e.g. rephrasing in passive form)
- *Projection Principle*: lexical properties preserved when forming new sentences from given ones (phrase structure rules projected from lexical rules)
- *Subjacency Principle*: transformation moves are "local" (don't move elements of phrases across more than one "bounding node" S=sentence and NP=noun phrase)

Examples of Parameters

- *Head-directionality*
- *Subject-side*
- *Pro-drop*
- *Null-subject*

Problems

- Interdependencies between parameters
- Diachronic changes of parameters in language evolution

Word Order and Parameters

• *Subject-side* parameter: positioning of the subject with respect to the head (specifier-head, head-specifier, and subject-initial, subject-medial, subject-final)

• Word Order: SOV, SVO, VSO, VOS, OVS, OSV

Word Orders	Percentage		
SOV	41.03%	Subject-initial	Specifier-Head
SVO	35.44%		
VSO	6.90%	Subject-medial	Head-Specifier
VOS	1.82%	Subject-final	
OVS	0.79%		
OSV	0.29%	Subject-medial	Specifier-Head

Very uneven distribution across world languages

Changes over time in Word Order

• Ancient Greek: switched from Homeric to Classical
- A. Taylor, *The change from SOV to SVO in Ancient Greek*, Language Variation and Change, 6 (1994) 1–37

• Sanskrit: different word orders allowed, but prevalent one in Vedic Sanskrit is SOV
- F.J. Staal, *Word Order in Sanskrit and Universal Grammar*, Springer, 1967

• English: switched from Old English (transitional between SOV and SVO) to Middle English (SVO)
- J. McLaughlin, *Old English Syntax: a handbook*, Walter de Gruyter, 1983.

• Word order distribution: a neuroscience explanation?

- D. Kemmerer, *The cross-linguistic prevalence of SOV and SVO word orders reflects the sequential and hierarchical representation of action in Broca's area*, Language and Linguistics Compass, 6 (2012) N.1, 50–66.

• Internal reasons for diachronic switch?

- F.Antinucci, A.Duranti, L.Gebert, *Relative clause structure, relative clause perception, and the change from SOV to SVO*, Cognition, Vol.7 (1979) N.2 145–176.

• knowledge of relation between parameters may give better explanations

Dependent parameters

- null-subject parameter: can drop subject

Example: among Latin languages, Italian and Spanish have null-subject (+), French does not (-)
it rains, piove, llueve, il pleut

- pro-drop parameter: can drop pronouns in sentences

不知道。喜欢吗?

Bù zhīdào. Xǐhuan ma?

- Pro-drop controls Null-subject

How many independent parameters?

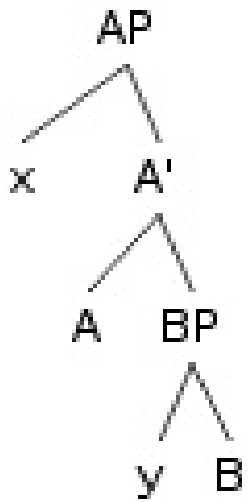
Government and Binding

- based on Principles and Parameters model of language

- if A and B are two nodes in a syntactic tree,

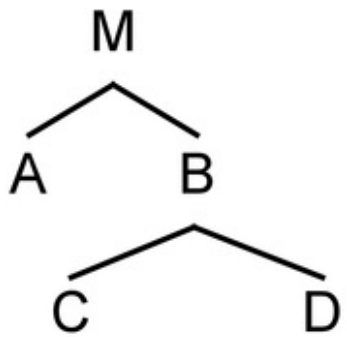
A *m-commands* B iff

- neither node dominates the other
- the maximal projection AP of A dominates B



A m-commands B, but B does not m-command A

• if X and Y are two nodes in a syntactic tree,
X *c-commands* Y (constituent command) iff
- neither node dominates the other
- the first node that dominates X also dominates Y



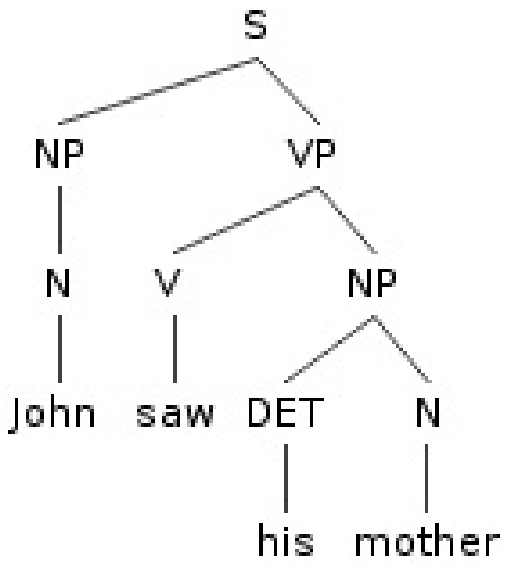
A and B c-command each other, C and D also, A also c-commands C and D

• A node X m-commands all nodes it c-commands, but also the nodes in XP

- node A governs B iff
- A is a governor (head of lexical category V,N,A,...)
- A m-commands B
- no barrier between A and B

- A barrier (between A and B) is a node X in a syntactic tree
- X c-commands B
- X does not c-command A

- Binding A binds B iff
- A c-commands B
- A and B are coreferential (refer to the same person)



in this sentence, "John" binds "his"

- These rules are used to test grammaticality of sentences

Minimalist Program (Chomsky, 1993)

• originally formulated as a program not a theory: guiding conceptual framework

• full mathematical formalization is only now being developed for the more recent formulation of Minimalism

• formulated within Principles and Parameters setting

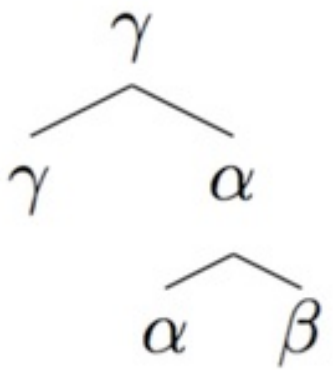
• postulates the existence of an underlying *simple computational structure* responsible for linguistic capability in the human mind (related to the idea of Universal Grammar)

• some minimality assumptions: *economy of representation* (sentence structure no more complicated than minimally required to satisfy constraints imposed by grammaticality); *economy of derivation* (transformations only occur if they make parts of sentence interpretable, e.g. disambiguation produced by inflection)

- Bare phrase structure: an explicitly derivational model of sentence building (as opposed to representational)
- Basic operations: merge and move
- Merge: $(\alpha, \beta) \mapsto \{\alpha, \{\alpha, \beta\}\}$ or $\{\beta, \{\alpha, \beta\}\}$

Example: (drink, water) $\mapsto$ { drink , { drink , water }}
Example: (cold, water) $\mapsto$ { water , { cold , water }}
The first merged "drink water" can be inserted in a sentence in place of "drink"; the second merge "cold water" can be inserted in place of "water"

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



• Move: moving parts of a sentence within the sentence

Example:
You are looking for someone
Whom are you looking for?

• Placeholder symbol (trace) for the moved element
(Whom) are you looking for (t)?
"Whom" and its trace symbol "(t)" are a *chain*,
similarly for other elements that change position

• Currently different approach: the "Move" operation and "Merge" are unified (see later in this class)

Next: Before introducing the new theory, a closer look at some of these parts of the historical development of Generative Linguistics