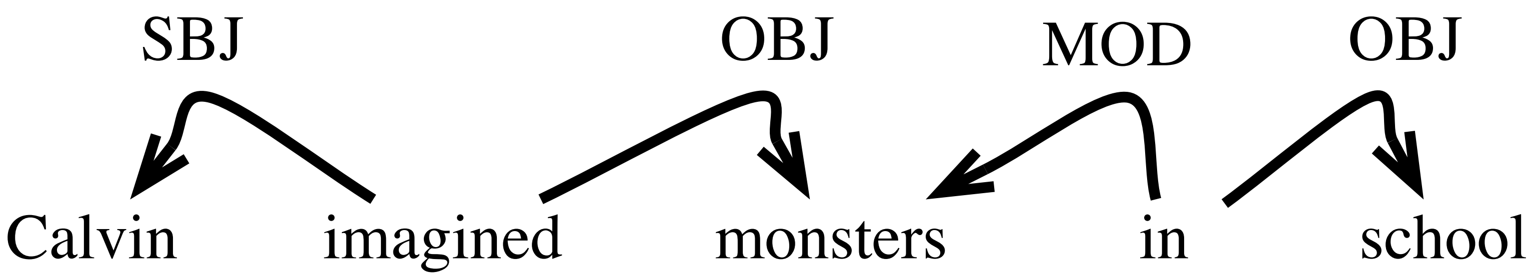
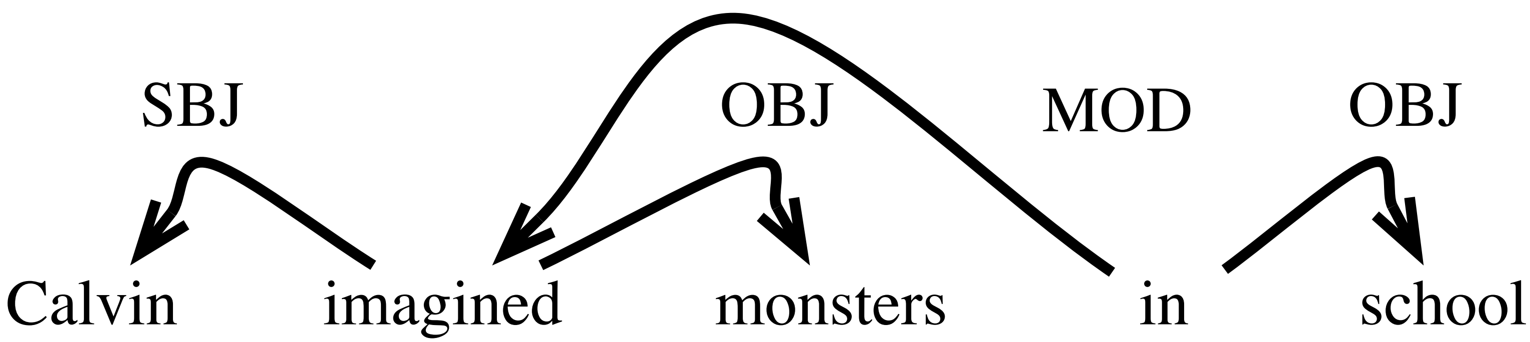
CMPT-413: Computational Linguistics

Anoop Sarkar
anoop@cs.sfu.ca
www.sfu.ca/~anoop/courses/CMPT-413-Spring-2003.html

Dependency Grammar





Dependency Grammar: (Tesnière, 1959), (Panini)

1	Calvin	2	SBJ
2	imagined	–	TOP
3	monsters	2	OBJ
4	in	{2,3}	MOD
5	school	4	OBJ

- If the dependencies are *nested* then DGs are equivalent (formally) to CFGs

  1. TOP(imagined) $\rightarrow$ SBJ(Calvin) imagined OBJ(monsters) MOD(in)

  2. MOD(in) $\rightarrow$ in OBJ(school)

- However, each rule is lexicalized (has a terminal symbol)

Categorial Grammar (Adjukiewicz, 1935)

Calvin	hates	mangoes
NP	(S\NP)/NP	NP
	S\NP	
	S	

- Also equivalent to CFGs
- Similar to DGs, each rule in CG is lexicalized

Natural Language and Complexity

- Formal language theory in computer science is a way to quantify computation
- From regular expressions to Turing machines, we obtain a hierarchy of recursion
- We can similarly use formal languages to describe the set of human languages
  Usually we abstract away from in the individual words in the language and concentrate on general aspects of the language

Natural Language and Complexity

- We ask the question: *Does a particular formal language describe some aspect of human language*
- Then we find out if that language **isn't** in a particular language class
- For example, if we abstract some aspect of human language to the formal language: $\{ww^R \mid \text{ where } w \in \{a,b\}^*, w^R \text{ is the reverse of } w\}$ we can then ask if it is possible to write a regular expression for this language
- If we can, then we can say that this particular example from human language does not go beyond regular languages. If not, then we have to go higher in the hierarchy (say, up to context-free languages)

The Chomsky Hierarchy

- **unrestricted** or **type-0** grammars, generate the *recursively enumerable* languages, automata equals *Turing machines*

- **context-sensitive** grammars, generate the *context-sensitive* languages, automata equals *Linear Bounded Automata*

- **context-free** grammars, generate the *context-free* languages, automata equals *Pushdown Automata*

- **regular** grammars, generate the *regular* languages, automata equals *Finite-State Automata*

The Chomsky Hierarchy: $G = (V, T, P, S)$ where, $\alpha, \beta, \gamma \in (N \cup T)^*$

- **unrestricted** or **type-0** grammars: $\alpha \rightarrow \beta$, such that $\alpha \neq \epsilon$
- **context-sensitive** grammars: $\alpha A \beta \rightarrow \alpha \gamma \beta$, such that $\gamma \neq \epsilon$
- **context-free** grammars: $A \rightarrow \gamma$
- **regular** grammars: $A \rightarrow a\ B$ or $A \rightarrow a$

Regular grammars: **right-linear CFG**: $L(G) = \{a^* b^* \mid n \geq 0\}$

$$\begin{array}{lcl} A & \rightarrow & a\ A \\ A & \rightarrow & \epsilon \\ A & \rightarrow & b\ B \\ B & \rightarrow & b\ B \\ B & \rightarrow & \epsilon \end{array}$$

Context-free grammars: $L(G) = \{a^n b^n \mid n \geq 0\}$

$$
\begin{array}{rcl}
S & \rightarrow & a\,S\,b \\
S & \rightarrow & \epsilon
\end{array}
$$

Context-sensitive grammars: $L(G) = \{a^n b^n \mid n \geq 1\}$

$$
\begin{aligned}
S &\rightarrow S\ B\ C \\
S &\rightarrow a\ C \\
a\ B &\rightarrow a\ a \\
C\ B &\rightarrow B\ C \\
B\ a &\rightarrow a\ a \\
C &\rightarrow b
\end{aligned}
$$

Context-sensitive grammars: $L(G) = \{a^n b^n \mid n \geq 1\}$

$$
\begin{array}{r}
S_1 \\
S_2\ B_1\ C_1 \\
S_3\ B_2\ C_2\ B_1\ C_1 \\
a_3\ C_3\ B_2\ C_2\ B_1\ C_1 \\
a_3\ B_2\ C_3\ C_2\ B_1\ C_1 \\
a_3\ a_2\ C_3\ C_2\ B_1\ C_1 \\
a_3\ a_2\ C_3\ B_1\ C_2\ C_1 \\
a_3\ a_2\ B_1\ C_3\ C_2\ C_1 \\
a_3\ a_2\ a_1\ C_3\ C_2\ C_1 \\
a_3\ a_2\ a_1\ b_3\ b_2\ b_1
\end{array}
$$

Unrestricted grammars: $L(G) = \{a^{2i} \mid i \geq 1\}$

$$\begin{array}{rcl}
S & \rightarrow & A\ C\ a\ B \\
C\ a & \rightarrow & a\ a\ C \\
C\ B & \rightarrow & D\ B \\
\mathbf{C\ B} & \rightarrow & \mathbf{E} \\
a\ D & \rightarrow & D\ a \\
A\ D & \rightarrow & A\ C \\
a\ E & \rightarrow & E\ a \\
\mathbf{A\ E} & \rightarrow & \epsilon
\end{array}$$

Unrestricted grammars: $L(G) = \{a^{2i} \mid n \geq 1\}$

$$\begin{array}{r} S \\ A\ C\ a\ B \\ A\ a\ a\ C\ B \\ A\ a\ a\ E \\ A\ a\ E\ a \\ A\ E\ a\ a \\ a\ a \end{array}$$

Unrestricted grammars: $L(G) = \{a^{2^i} \mid i \geq 1\}$

- A and B serve as left and right end-markers for sentential forms (derivation of each string)

- C is a marker that moves through the string of a's between A and B, doubling their number using $C\ a \to a\ a\ C$

- When C hits right end-marker B, it becomes a D or E by $C\ B \to D\ B$ or $C\ B \to E$

- If a D is chosen, that D migrates left using $a\ D \to D\ a$ until left end-marker A is reached

- At that point D becomes C using $A\ D \rightarrow A\ C$ and the process starts over

- Finally, E migrates left until it hits left end-marker A using $a\ E \rightarrow E\ a$

- Note that $L(G) = \{a^{2i} \mid i \geq 1\}$ can also be written as a context-sensitive grammar, but consider G', where $L(G') = \{a^{2i} \mid i \geq 0\}$ can only be an unrestricted grammar. Note that $a^0 = \epsilon$. Why is this true?

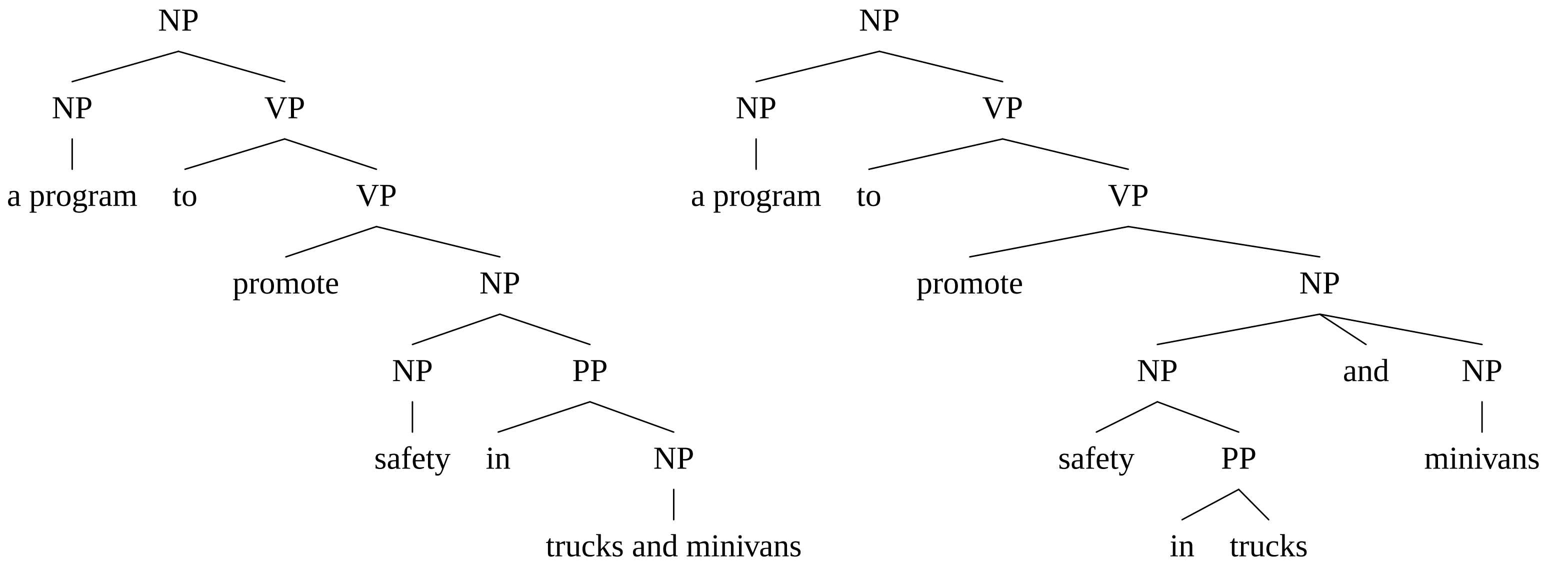
Strong vs. Weak Generative Capacity

- **Weak generative capacity** of a grammar is the set of strings or the language, e.g. $0^n 1^n$ for $n \geq 0$
- **Strong generative capacity** is the set of structures (usually the set of trees) provided by the grammar
- Let's ask the question: is the set of human languages contained in the set of regular languages?

Strong vs. Weak Generative Capacity

- If we consider strong generative capacity then the answer is somewhat easier to obtain
- For example, do we need to combine two non-terminals to provide the semantics?
- Or do we need nested dependencies?

Strong vs. Weak Generative Capacity



Strong vs. Weak Generative Capacity

- However, strong generative capacity requires a particular grammar and a particular linguistics theory of semantics or how meaning is assigned (in steps or compositionally)

- So, the stronger claim will be that some aspect of human language when you consider *weak* generative capacity is not regular

- This is quite tricky: consider $L_1 = \{a^n b^n\}$ is context-free but $L_2 = \{a^* b^*\}$ is regular and $L_1 \subset L_2$: so you could cheat and pick some subset of the language which won't prove anything

- Furthermore, the language should be *infinite*

Strong vs. Weak Generative Capacity

- Also, if we consider the *size* of a grammar then also the answer is easier to obtain ($*$joyable, $*$richment). The CFG is more *elegant* and smaller than the equivalent regular grammar:

$$
\begin{array}{rcl}
V & \rightarrow & X \\
A & \rightarrow & X\ \textit{-able} \mid X\ \textit{-ment} \\
X & \rightarrow & \textit{en-}\ NA \\
NA & \rightarrow & \textit{joy} \mid \textit{rich}
\end{array}
$$

- This is an engineering argument. However, it is related to the problem of describing the human learning process. Certain aspects of language are learned all at once not individually for each case.
  e.g., learning *enjoyment* automatically if *enrichment* was learnt

Is Human Language a Regular Language

- Consider the following set of English sentences (strings)
  - S = If S_1 then S_2
  - S = Either S_3, or S_4
  - S = The man who said S_5 is arriving today
- Map *If, then* $\rightarrow a$ and *either, or* $\rightarrow b$. This results in strings like $abba$ or $abaaba$ or $abbaabba$
- $L = \{ww^R \mid \text{ where } w \in \{a, b\}^*, w^R \text{ is the reverse of } w\}$

Human Language is not a Regular Language

- Is $L = ww^R$ a regular language? To show something is *not* a regular language, we use the **pumping lemma**: for any infinite set of strings generated by a FSA if you consider a long enough string from this set, there has to be a loop which visits the same state at least twice

- Thus, in a regular language L, there are strings x, y, z such that xy^nz for $n \geq 0$ where $y \neq \epsilon$

- Let L' be the intersection of L with aa^*bbaa^*. Recall that RLs are closed under intersection, so L' must also be a RL. $L' = a^nbba^n$
  For any choice of y (consider a^i or b^i or a^ib or ba^i) the pumping lemma leads to the conclusion that L' is **not** regular.

Human Language is not a Regular Language

- Another example, also from English, is the set of center embedded structures
  Think of $S \rightarrow a\ S\ b$ and the nested dependencies $a_1 a_2 a_3 b_3 b_2 b_1$

- Center embedding in English:
  the shares that the broker recommended were bought $\Rightarrow N_1 N_2 V_2 V_1$
  the moment when the shares that the broker recommended were bought has passed $\Rightarrow N_1 N_2 N_3 V_3 V_2 V_1$

- Can you come up with an example that has four verbs and corresponding number of nouns?
  cf. *The Embedding* by Ian Watson

Human Competence vs. Human Performance

- What if no more than 3 or 4 center embedding structures are possible? Then the language is finite, so the language is no longer strictly context-free

- The common assumption made is that human competence is represented by the context-free grammar, but human performance suffers from memory limitations which can be simulated by a simpler mechanism

- The arguments about elegance, size and the learning process in humans also apply in this case

Human Language is not a Context-Free Language

- Two approaches as before: consider **strong** and **weak** generative capacity

- For strong generative capacity, if we can show crossing dependencies in a language then no CFG can be written for such a language. Why?

- Quite a few major languages spoken by humans have crossed dependencies:
  Dutch (Bresnan et al., 1982), Swiss German, Tagalog, among others.

Human Language is not a Context-Free Language

- Swiss German:

...	mer	em Hans	es huus	hälfed	aastriiche
...	we	Hans-DAT	the house-ACC	helped	paint
		N_1	N_2	V_1	V_2

... *we helped Hans paint the house*

- Analogous structures in English (PRO is a empty pronoun subject):

Eng:	S_1 = we $[_{V_1}$ helped$]$ $[_{N_1}$ Hans$]$ (to do) $[_{S_2} \ldots]$
SwGer:	S_1 = we $[_{N_1}$ Hans$]$ $[_{S_2} \ldots [_{V_1}$ helped$] \ldots]$
Eng:	S_2 = PRO(ϵ) $[_{V_2}$ paint$]$ $[_{N_2}$ the house$]$
SwGer:	S_2 = PRO(ϵ) $[_{N_2}$ the house$]$ $[_{V_2}$ paint$]$
Eng:	$S_1 + S_2$ = we helped$_1$ Hans$_1$ PRO(ϵ) paint$_2$ the house$_2$
SwGer:	$S_1 + S_2$ = we Hans$_1$ PRO(ϵ) the house$_2$ helped$_1$ paint$_1$

Human Language is not a Context-Free Language

- Weak generative capacity of human language being greater than context-free was much harder to show. (Pullum, 1982) was a compendium of all the failed efforts so far.

- (Shieber, 1985) and (Huybregts, 1984) showed this using examples from Swiss-German:

mer	d'chind	em Hans	es huus	lönd	hälfed	aastriiche
we	the children-ACC	Hans-DAT	the house-ACC	let	helped	paint
w	a	b	x	c	d	y
	N_1	N_2	N_3	V_1	V_2	V_3

. . . *we let the children help Hans paint the house*

- Let this set of sentences be represented by a language L (mapped to symbols w, a, b, x, c, d, y)

- Do the usual intersection with a regular language: $wa^*b^*xc^*d^*y$ to obtain $L' = wa^m b^n xc^m d^n y$

- The pumping lemma for CFLs [Bar-Hillel] states that if a string from the CFL can be written as $wuxvy$ for $u, v \neq \epsilon$ and $wuxvy$ is long enough then $wu^n xv^n y$ for $n \geq 0$ is also in that CFL.

- The pumping lemma for CFLs shows that L' is not context-free and hence human language is not even *weakly* context-free

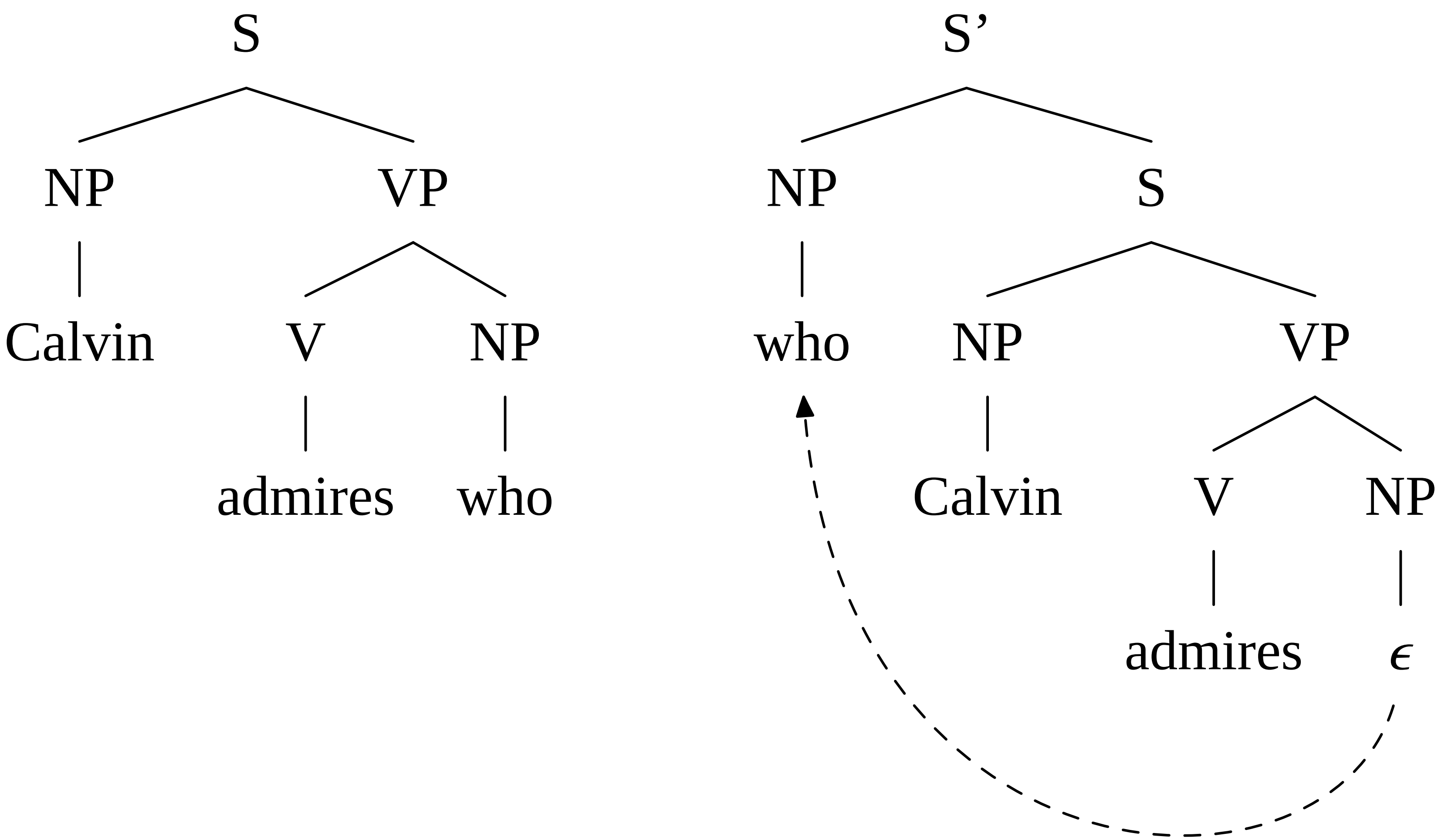
Transformational (Movement) Grammars

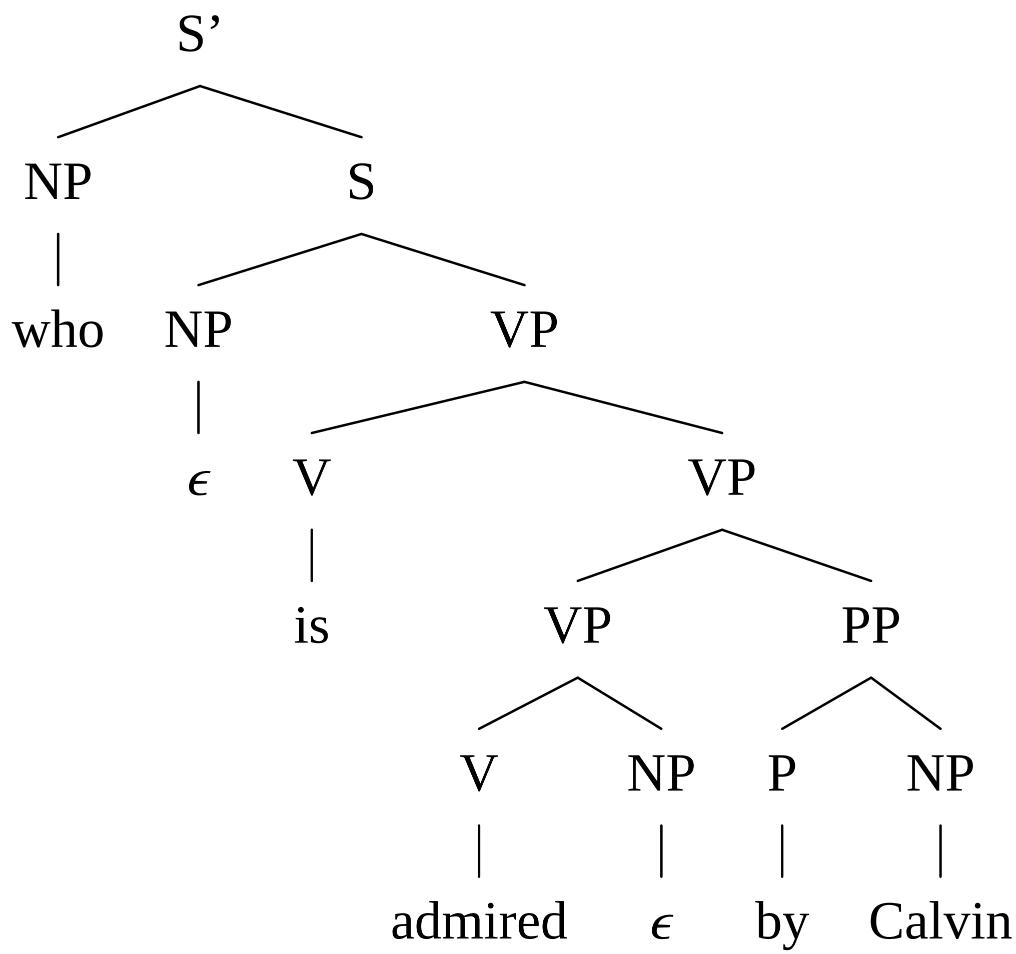
Note: **not** related to Transformation-Based Learning

- As we saw showing strong generative capacity beyond context-free was quite easy: all we needed was crossed dependencies to link verbs with their arguments.
- Linguists care about strong generative capacity since it provides the means to compute meanings using grammars.
- Linguists also want to express generalizations (cf. the morphology example: $*$joyment, $*$richment)

Transformational (Movement) Grammars

Calvin	admires	Hobbes .
Hobbes is	admired	by Calvin .
Who does Calvin	admire	?
Who	admires	Hobbes ?
Who does Calvin believe	admires	Hobbes ?
The stuffed animal who	admires	Hobbes is a genius .
The stuffed animal who Calvin	admires	is imaginative .
Who is	admired	by Calvin ?
The stuffed animal who is	admired	by Calvin is a genius .
Who is Hobbes	admired	by ?
The stuffed animal who Hobbes is	admired	by is imaginative .
Calvin seems to	admire	Hobbes .
Calvin is likely to seem to	admire	Hobbes .
Who does Calvin think I believe Hobbes	admires	?


S
NP
VP
Calvin
V
NP
admires
who
S'
NP
S
who
NP
VP
Calvin
V
NP
admires
ϵ


S'
NP
S
who
NP
VP
ϵ
V
VP
is
VP
PP
V
NP
P
NP
admired
ϵ
by
Calvin

- **context-sensitive** grammars: 0^i, i is not a prime number and $i > 0$

- **indexed** grammars: $0^n 1^n 2^n \ldots m^n$, for any fixed m and $n \geq 0$

- **tree-adjoining** grammars (TAG), **linear-indexed** grammars (LIG), **combinatory categorial** grammars (CCG): $0^n 1^n 2^n 3^n$, for $n \geq 0$

- **context-free** grammars: $0^n 1^n$ for $n \geq 0$

- **deterministic context-free** grammars: $S' \rightarrow S\ c$, $S \rightarrow S\ A \mid A$, $A \rightarrow a\ S\ b \mid ab$: the language of ”balanced parentheses”

- **regular** grammars: $(0|1)^* 00 (0|1)^*$

Language	*Automaton*	*Grammar*	*Recognition*	*Dependency*
Recursively Enumerable Languages	Turing Machine	Unrestricted Baa → A	Undecidable	Arbitrary
Context-Sensitive Languages	Linear-Bounded	Context-Sensitive At → aA	NP-Complete	Crossing
Context-Free Languages	Pushdown (stack)	Context-Free S → gSc	Polynomial	Nested
Regular Languages	Finite-State Machine	Regular A → cA	Linear	Strictly Local

Given grammar G and input x, provide algorithm for: Is $x \in L(G)$?

- **unrestricted**: undecidable (movement grammars, feature structure unification)
- **context-sensitive**: NSPACE[n] – linear non-deterministic space
- **indexed** grammars: NP-Complete (restricted feature structure unification)
- **tree-adjoining** grammars (TAG), **linear-indexed** grammars (LIG), **combinatory categorial** grammars (CCG), **head** grammars: $\mathcal{O}(n^6)$
- **context-free**: $\mathcal{O}(n^3)$
- **deterministic context-free**: $\mathcal{O}(n)$
- **regular** grammars: $\mathcal{O}(n)$

Which class corresponds to human language?

Tree Adjoining Grammars

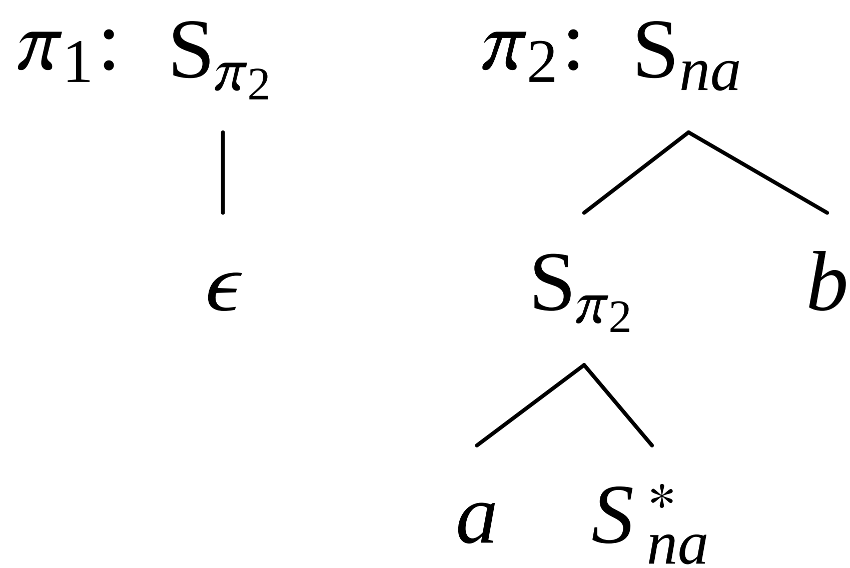
- Intuition we have is that the trees produced after parsing are important for computing the meaning. So instead of building the trees using context-sensitive rules like $\alpha A \beta \rightarrow \alpha \gamma \beta$, build in the context-sensitive part into trees

- Tree-adjoining grammar $G = (S, V, T, I, A)$ where
  - V is the set of non-terminal symbols
  - T is the set of terminal symbols
  - I is a set of non-recursive (terminal) trees

- A is the set of recursive (non-terminal) trees
- S is the set of start trees where $S \subseteq I$,
- $I \cup A$ is the set of *elementary trees*

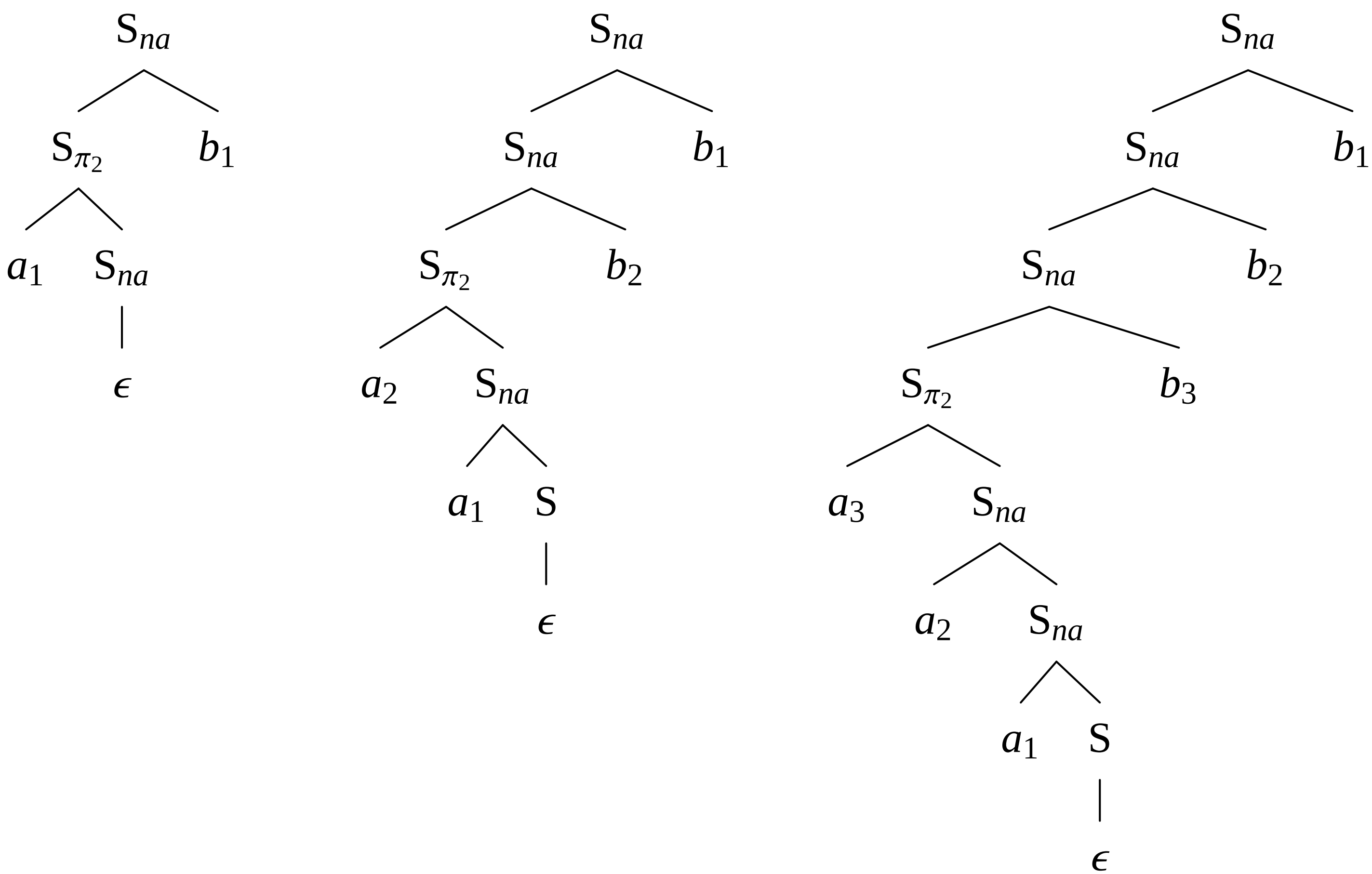
- Sits between context-free grammars and context-sensitive grammars
- Handles all the weak and strong cases used to argue for the non-context-free nature of language
- *Simple* handling of crossed and nested dependencies – compare with context sensitive grammars

Crossing Dependencies in TAG: $a_2 a_1 b_2 b_1$



$$G : (T = \{a, b, \epsilon\}, V = \{S\}, I = \{\pi_1\}, A = \{\pi_2\}, S = \{\pi_1\})$$

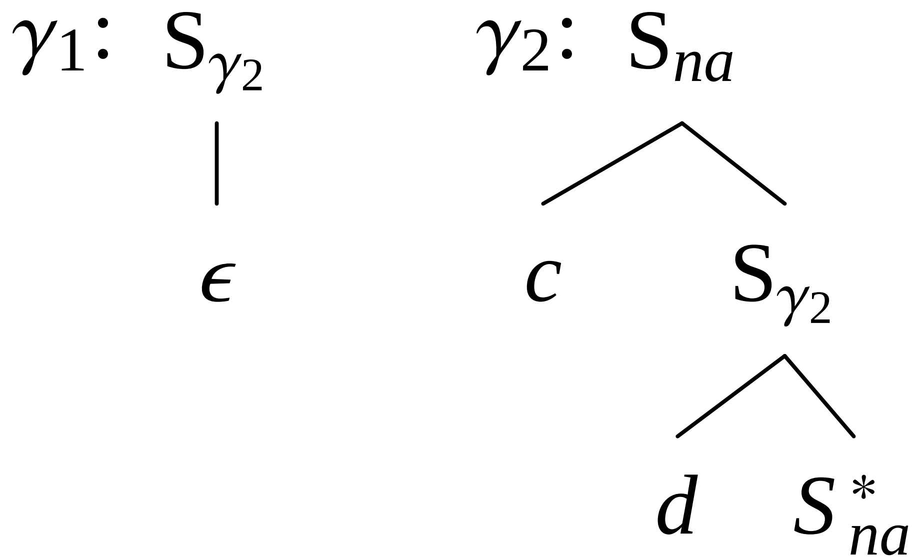
Crossing Dependencies in TAG: *Derived Tree* $a_3a_2a_1b_3b_2b_1$



Crossing Dependencies in TAG: *Derivation Tree*

$$
\begin{array}{c}
\pi_1(\epsilon) \\
| \\
\pi_2(a_1, b_1)
\end{array}
\qquad
\begin{array}{c}
\pi_1(\epsilon) \\
| \\
\pi_2(a_1, b_1) \\
| \\
\pi_2(a_2, b_2)
\end{array}
\qquad
\begin{array}{c}
\pi_1(\epsilon) \\
| \\
\pi_2(a_1, b_1) \\
| \\
\pi_2(a_2, b_2) \\
| \\
\pi_2(a_3, b_3)
\end{array}
$$

Nested Dependencies in TAG: $c_1 c_2 d_2 d_1$



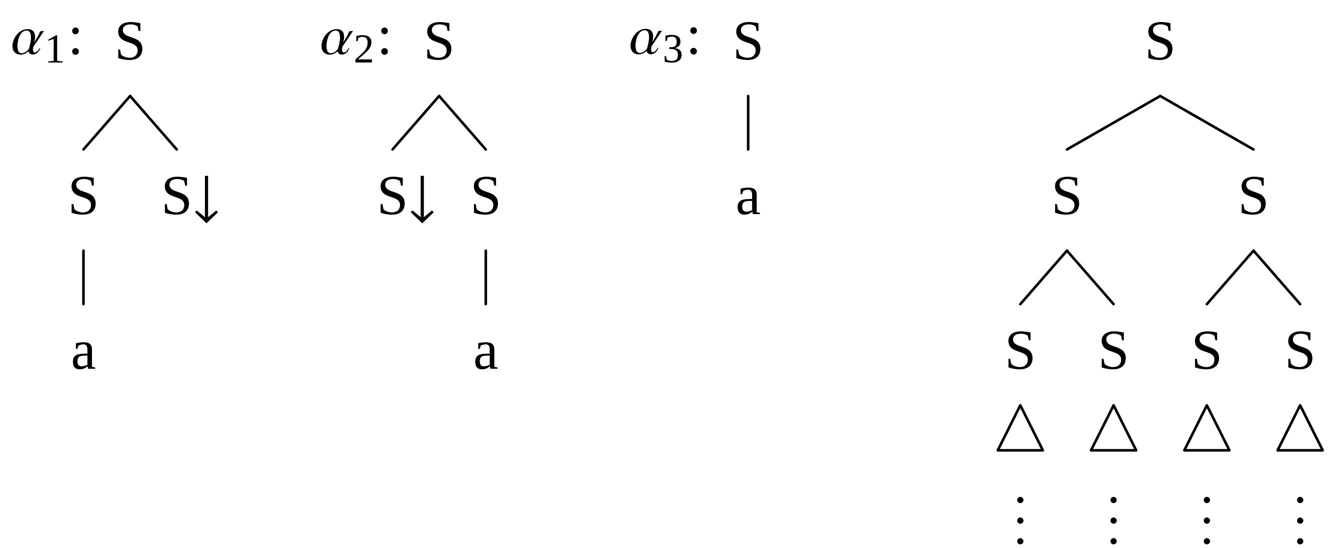
$$G : (T = \{c, d, \epsilon\}, V = \{S\}, I = \{\gamma_1\}, A = \{\gamma_2\}, S = \{\gamma_1\})$$

What happens if we put together a new grammar with trees $\pi_1, \pi_2, \gamma_1, \gamma_2$?

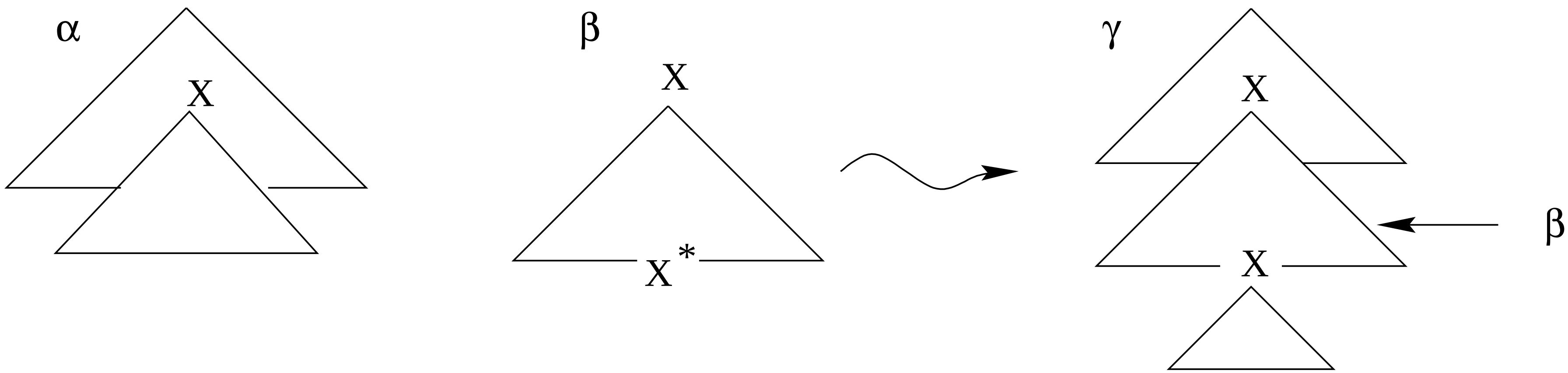
Lexicalization of Context-Free Grammars

- CFG G: $(r_1)\ S \rightarrow\ S\ S \quad (r_2)\ S \rightarrow\ a$

- Tree-substitution Grammar G':

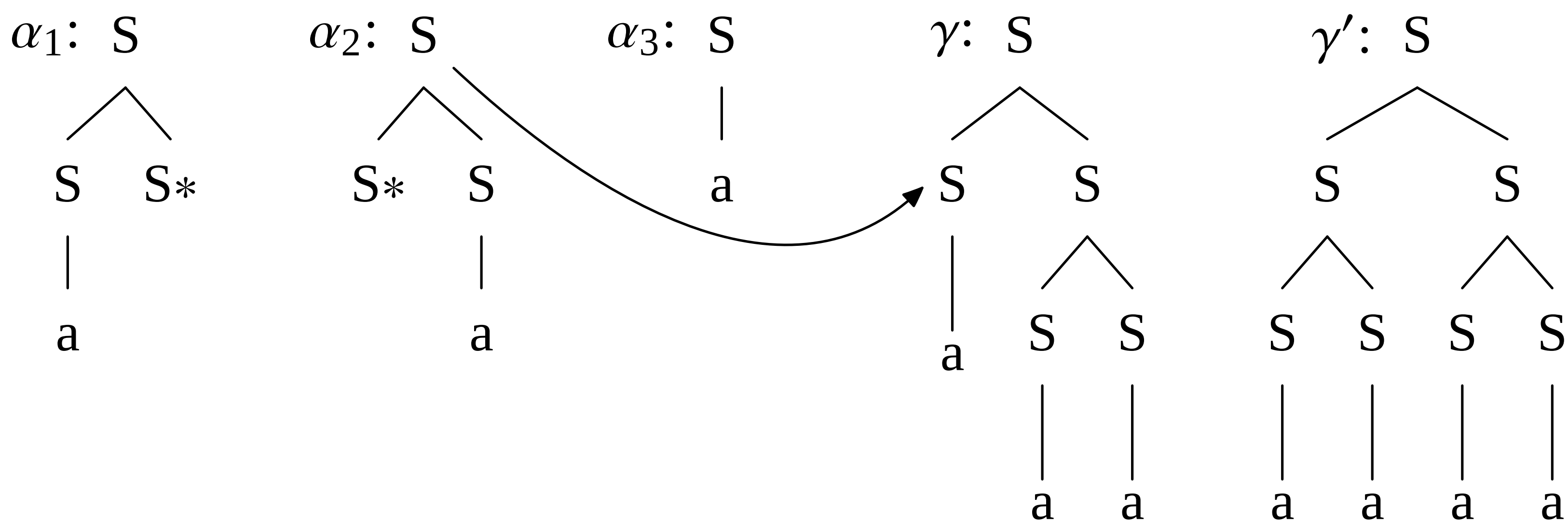


Lexicalization of Context-Free Grammars

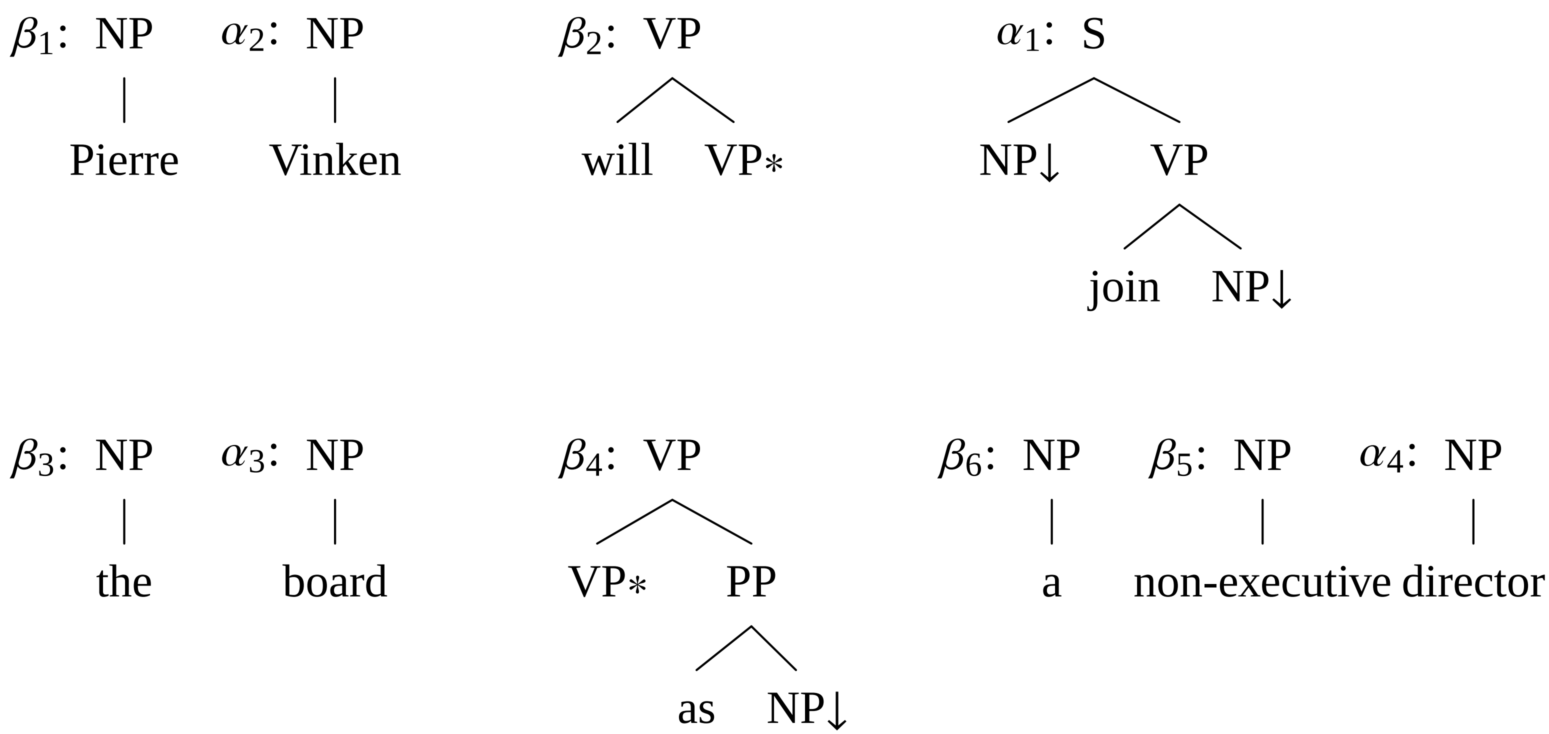


Lexicalization of Context-Free Grammars

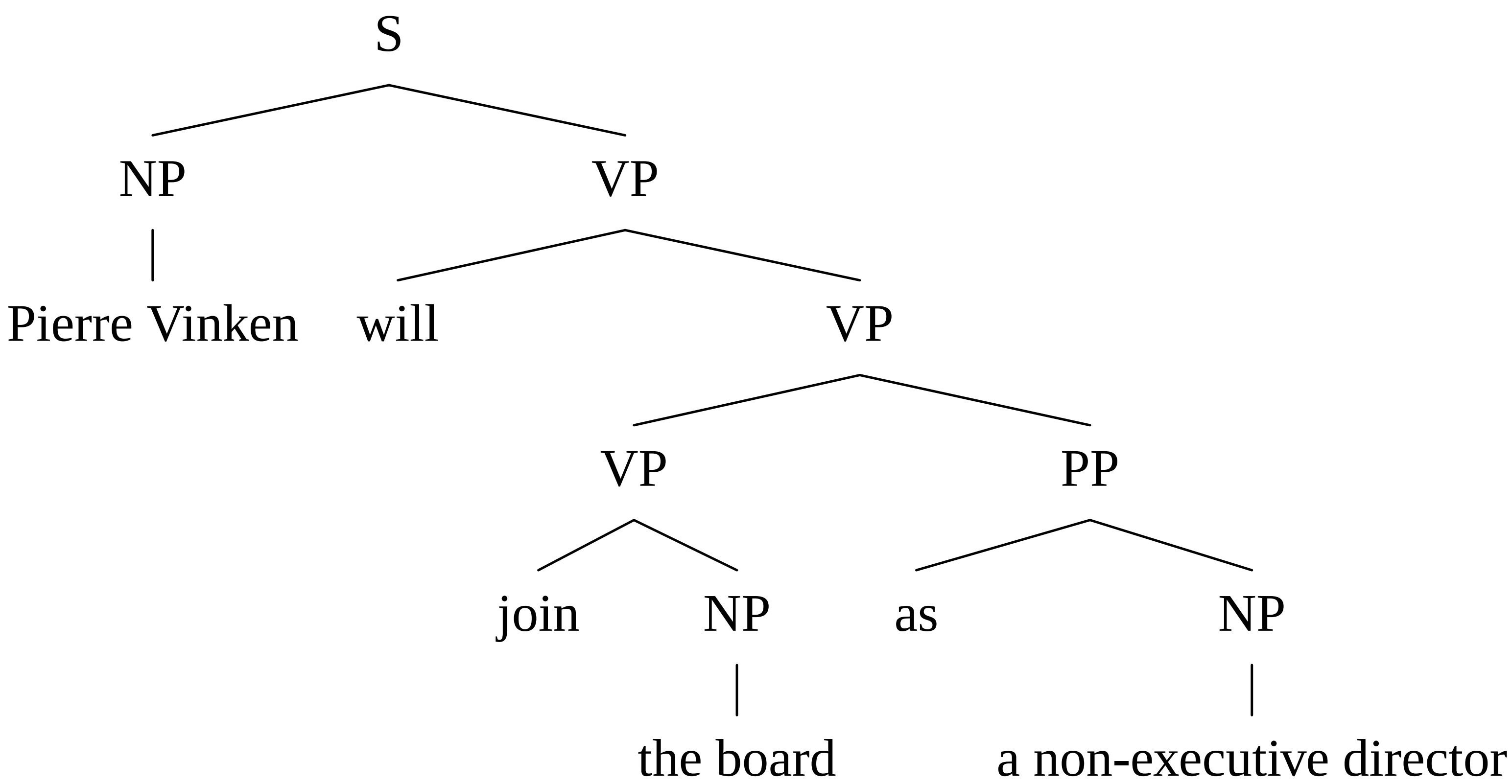
- CFG G: $(r_1)\ S \rightarrow\ S\ S$ $\quad (r_2)\ S \rightarrow\ a$
- Tree-adjoining Grammar G'':



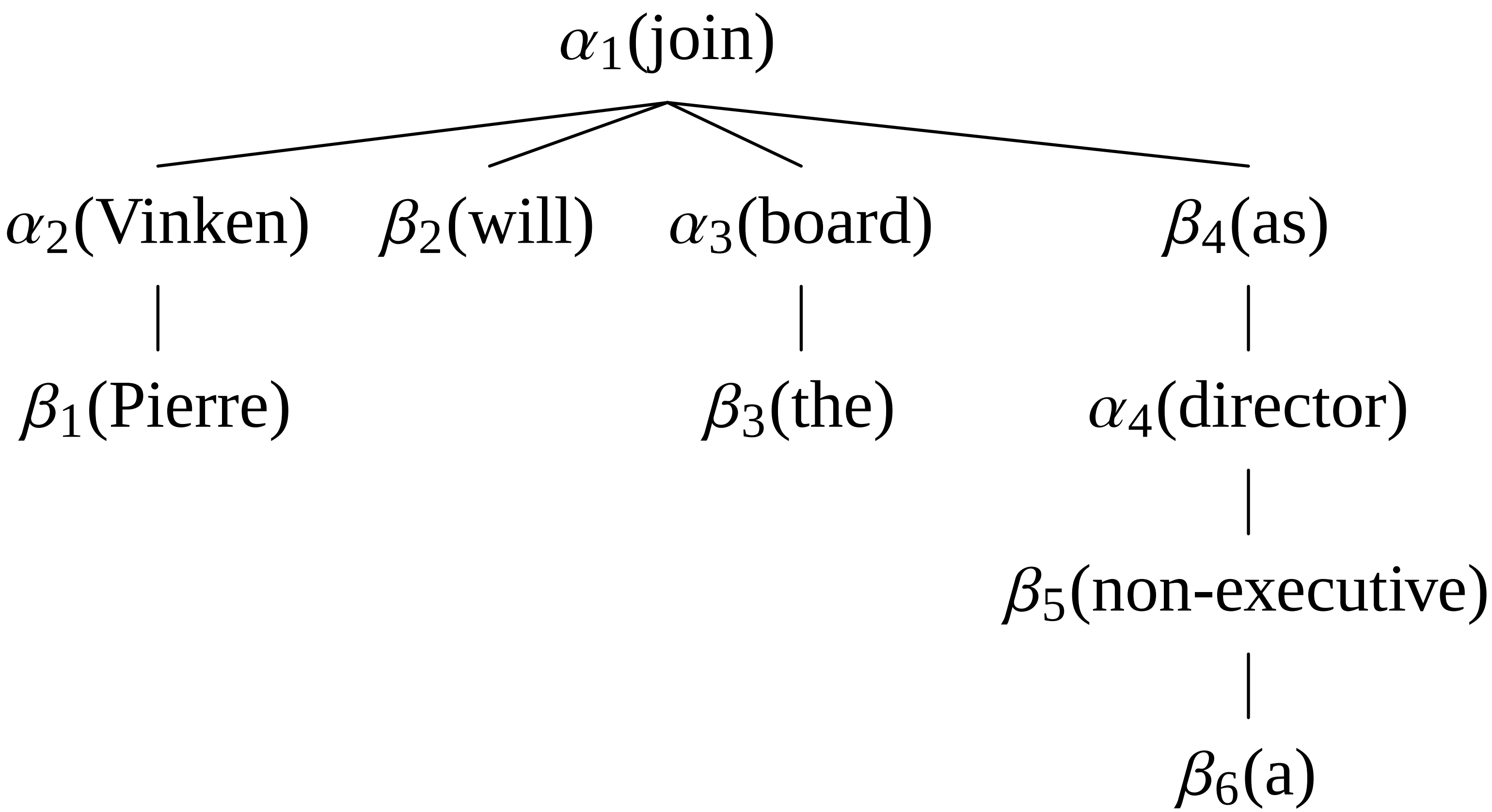
Lexicalized Tree Adjoining Grammars



Derived Tree



Derivation Tree



Derivation Trees for TAG

- Provides the *history* of how the trees were put together
- Compare with dependency grammars? What are the similarities and differences?