Formal Language, Natural Language and Tree Adjoining Grammars

Anoop Sarkar
anoop@cs.sfu.ca

Overview

- Tree Adjoining Grammars

$$\text{TAGs}\begin{cases}\text{Weak vs. Strong Generative Capacity}\\ \text{Natural Language and Complexity}\\ \text{Lexicalization of Grammars}\end{cases}$$

- TAGs for Natural Languages

Strong vs. Weak Generative Capacity

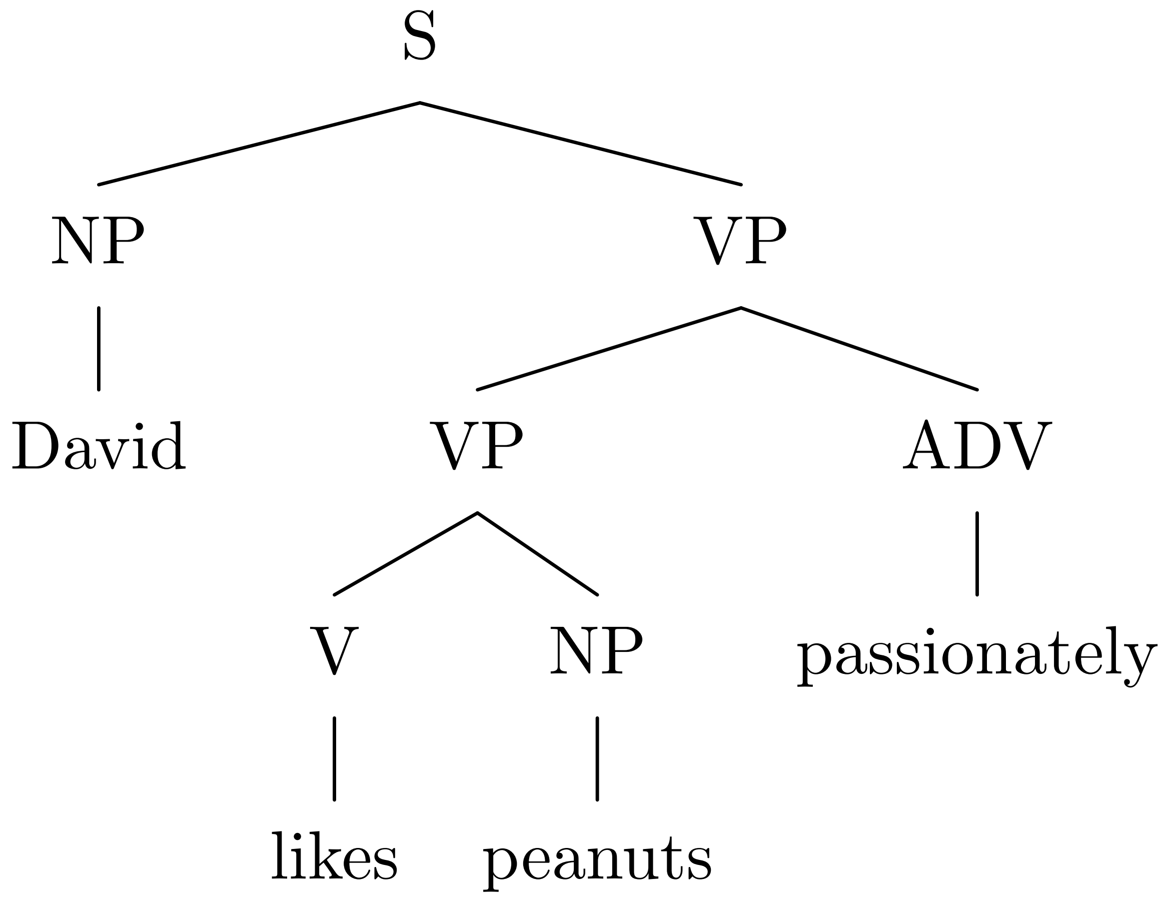
- **Weak generative capacity** of a grammar is the set of strings or the language, e.g. 0^n1^n for $n \geq 0$
- **Strong generative capacity** is the set of structures (usually the set of trees) provided by the grammar

Strong vs. Weak Generative Capacity

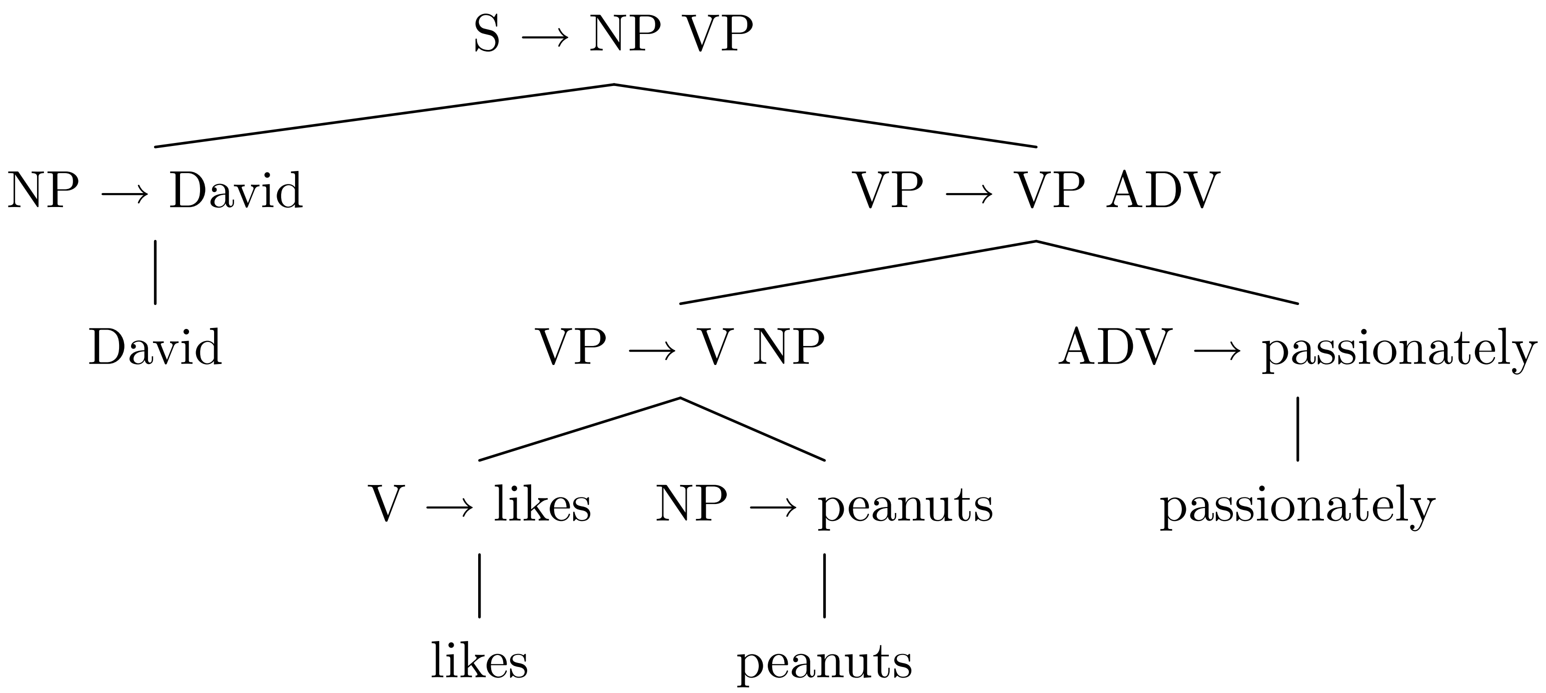
$$\begin{aligned}
S &\rightarrow NP\ VP && (1)\\
VP &\rightarrow V\ NP \mid VP\ ADV && (2)\\
NP &\rightarrow David \mid peanuts && (3)\\
V &\rightarrow likes && (4)\\
ADV &\rightarrow passionately && (5)
\end{aligned}$$

$$L(G) = \{\text{David likes peanuts, David likes peanuts passionately}\}$$

Derived Tree/Parse Tree

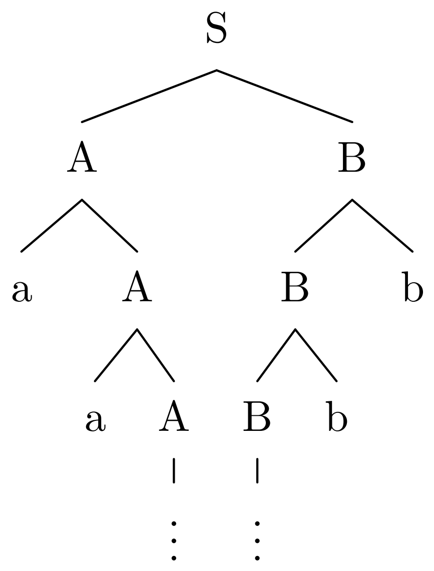


Derivation Tree



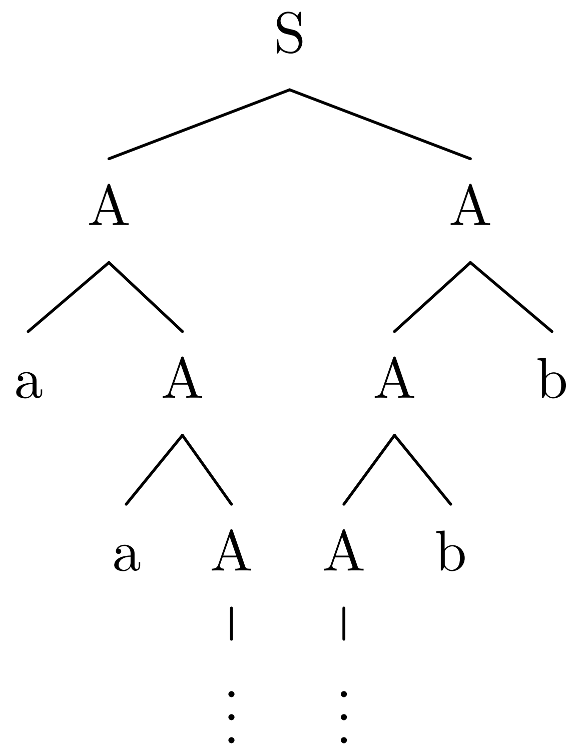
Tree Sets: Given CFG, G, let $T(G)$ be the set of derived trees

$$\begin{array}{rcl} S & \rightarrow & A\ B \\ A & \rightarrow & a\ A \mid a \\ B & \rightarrow & B\ b \mid b \end{array}$$

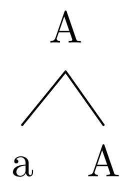


Tree set T shown here, $T = T(G)$

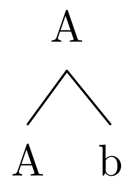
Tree Sets: Consider T' below. There is no CFG G such that $T(G) = T'$: Why?



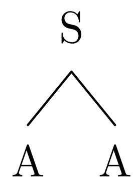
Local Tree Predicates

$A \rightarrow$ 

if there is a right tree context A

$A \rightarrow$ 

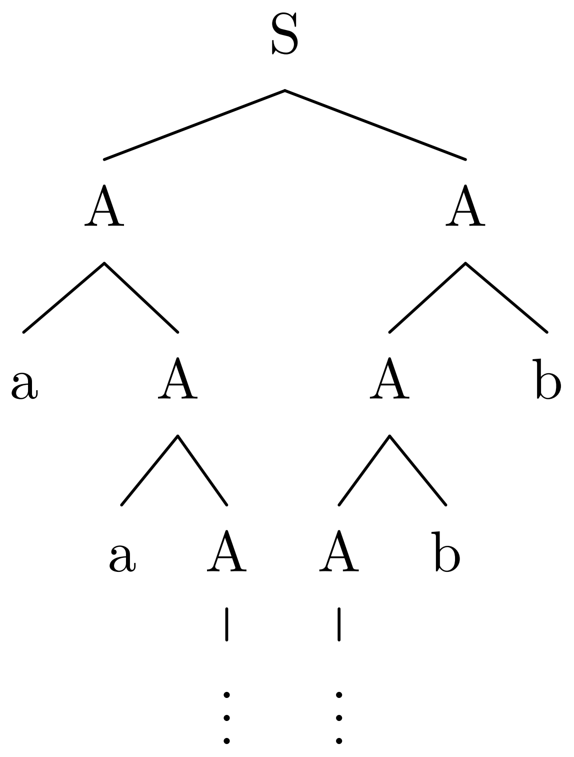
if there is a left tree context A

$S \rightarrow$ 

no context needed

Apply the predicates to trees and we can accept or reject trees. That is, we can *recognize* a tree set. Just like a grammar *recognizes* a set of strings, these predicates can recognize a set of trees.

Tree Sets



However, T' can be recognized by a finite-state bottom-up tree automaton

Tree Sets of CFGs and Recognizable Sets

- Recognizable Sets: Tree sets recognized by finite-state bottom-up tree automata: analogs of regular sets

- Strings sets of recognizable sets are context-free languages

- Recognizable sets are the same as tree sets of CFGs closed under relabelling homomorphisms (Thatcher, 1967)

- Additional strong generative power of recognizable sets by introducing local tree predicates

Tree Sets of CFGs and Recognizable Sets

- Surprisingly, even context-sensitive grammars when interpreted as local tree predicates can only *generate* the recognizable sets

- Local sets = Tree sets for CFGs; Recognizable sets = Local sets plus relabeling of the nodes

- Hence, CSGs when interpreted as “filters” on trees can only produce context-free languages (Peters and Ritchie, 1969; McCawley, 1967)

- Result can be strengthened to boolean combinations of CSG local tree predicates (Joshi, Levy and Yueh, 1972; Rogers, 1997)

Overview

- Tree Adjoining Grammars

$$\text{TAGs}\begin{cases}\text{\sout{Weak vs. Strong Generative Capacity}}\\ \text{Natural Language and Complexity}\\ \text{Lexicalization of Grammars}\end{cases}$$

- TAGs for Natural Languages

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)

Strong vs. Weak Generative Capacity

- If we consider strong generative capacity then the answer is somewhat easier to obtain

- For example, do we need nested dependencies to compute the semantics in a natural way?

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

- 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?

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 crossing dependencies:
  e.g Dutch (Bresnan et al., 1982), Swiss German (Shieber, 1984), Tagalog (Rambow and MacLachlan, 2002)

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*

- Clear case of crossing dependencies: so cannot be handled with CFGs.

The Chomsky Hierarchy: $G = (N, T, P, S)$ 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 $A \rightarrow a$

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

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

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

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

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 $\alpha A \beta \rightarrow \alpha \gamma \beta$, we can build in the context-sensitive part into **elementary trees** which is used to build larger trees.
  $\Rightarrow$ *Elementary trees are just like the rules in a context-free grammar.*

- The elementary trees become the basic units which can be used to **rewrite** non-terminal nodes inside elementary trees.
  $\Rightarrow$ *This rewriting is analogous to expanding a non-terminal in a context-free grammar.*

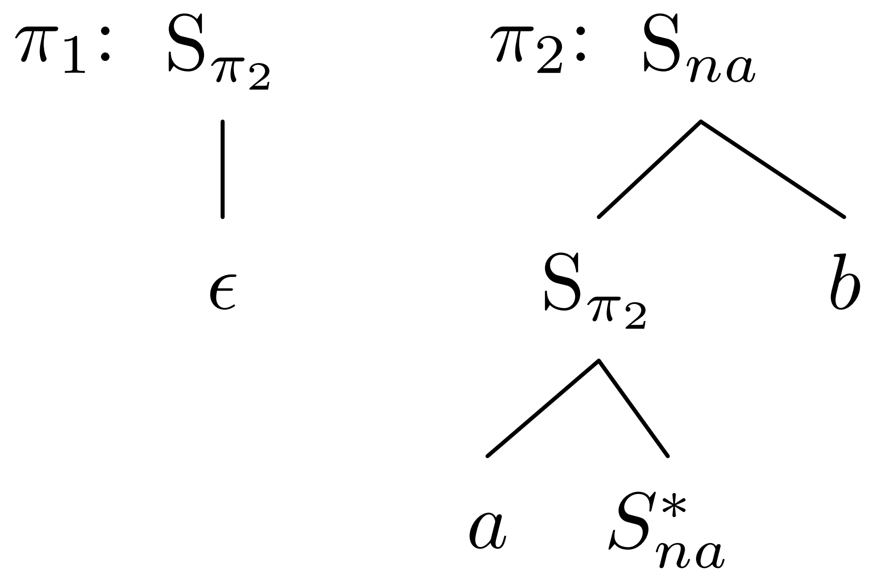
Tree Adjoining Grammar: Formal Definition

- Tree adjoining grammar $G = (S, N, T, I, A)$ where
  - N 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*

Tree Adjoining Grammars

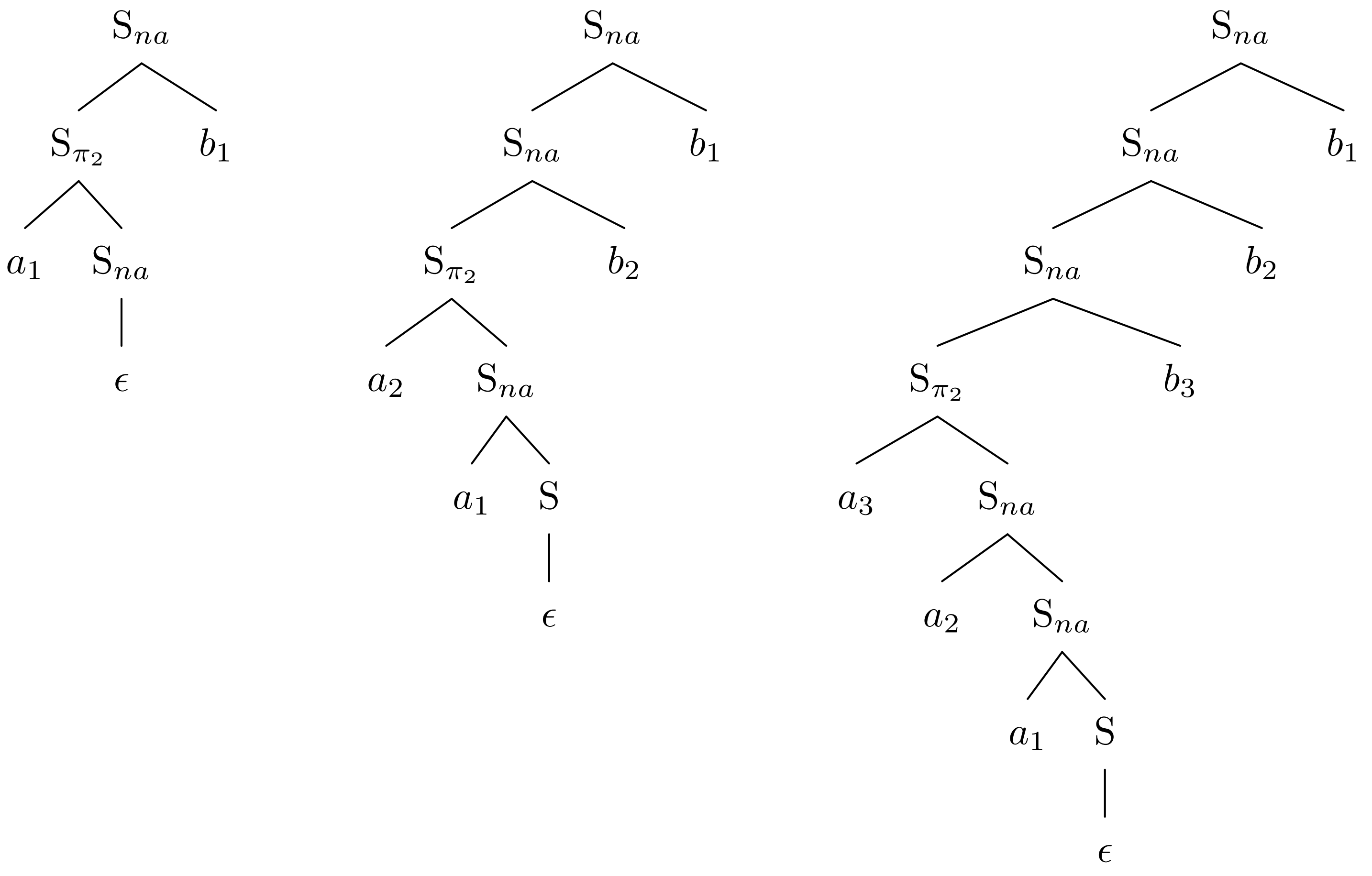
- Sits between context-free grammars and context-sensitive grammars
- Handles all the weak and strong cases used to argue for the non-context-free nature of language
- *Simple* handling of crossed and nested dependencies – compare with context sensitive grammars

Crossing Dependencies in TAG: $a_2 a_1 b_2 b_1$



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

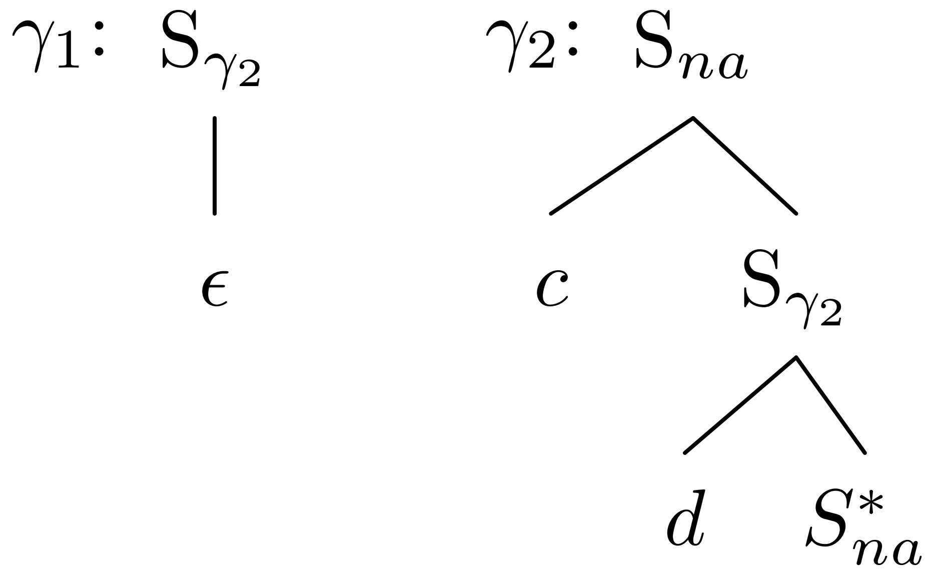
Crossing Dependencies in TAG: *Derived Tree* $a_3a_2a_1b_3b_2b_1$



Crossing Dependencies in TAG: *Derivation Tree*

$$\begin{tikzcd}
\pi_1(\epsilon) \arrow[d, dash] & \pi_1(\epsilon) \arrow[d, dash] & \pi_1(\epsilon) \arrow[d, dash] \\
\pi_2[0](a_1, b_1) & \pi_2[0](a_1, b_1) \arrow[d, dash] & \pi_2[0](a_1, b_1) \arrow[d, dash] \\
 & \pi_2[00](a_2, b_2) & \pi_2[00](a_2, b_2) \arrow[d, dash] \\
 & & \pi_2[00](a_3, b_3)
\end{tikzcd}$$

Nested Dependencies in TAG: $c_1 c_2 d_2 d_1$



γ_1: S_{γ_2}
ϵ
γ_2: S_{na}
c
S_{γ_2}
d
S^*_{na}

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

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

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

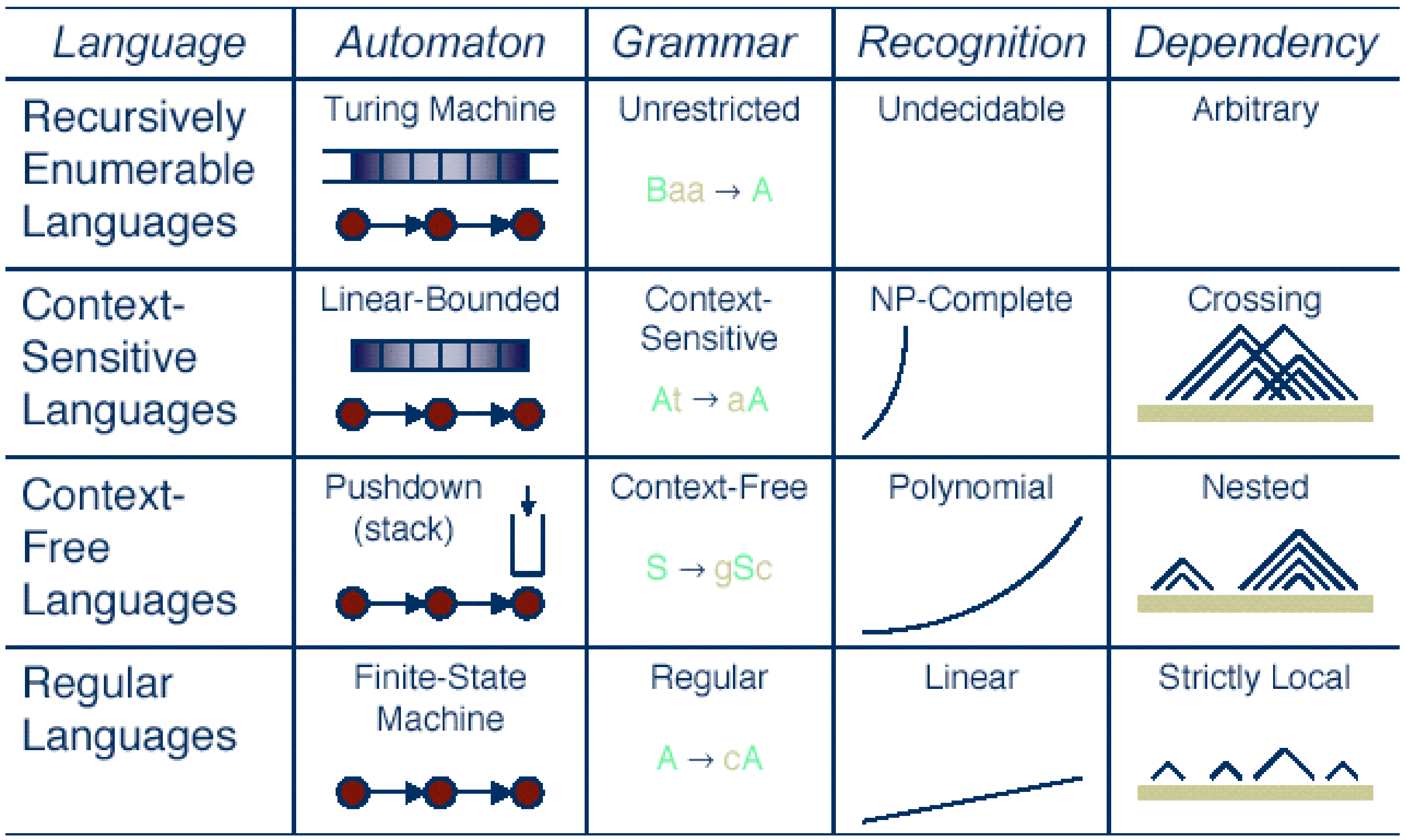
- **indexed** grammars: $0^n 1^n 2^n \ldots m^n$, for any fixed m and $n \geq 0$

- **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

TAG is between CSL and CFG:

has Polynomial time recognition and handles many crossing dependencies.

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

- **unrestricted**: undecidable (grammars using 'movement', 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

- Membership is in P ($\mathcal{O}(n^6)$)
- TALs are closed under union, concatenation, Kleene closure, h, h^{-1}, intersection with RLs and regular substitution. There is also a pumping lemma for TALs (proofs in (VijayShanker, 1987))
  Tree Adjoining Languages are a full abstract family of languages (AFL)
- No equivalent of Ogden's Lemma. TALs are not closed under intersection, intersection with CFLs and complementation.
- TAGs, LIGs, CCGs, HGs (see previous slide) were shown by TAG researchers to be **weakly equivalent** (VijayShanker, 1987; Weir, 1988)

Computationally Constrained Grammar Formalisms

- Why bother with computationally constrained grammar formalisms?
- Perhaps linguists can analyze language free of any thoughts about computational efficiency, using any system that can elegantly explain the data
- And then if at all necessary this analysis can be translated to a computationally constrained formalism

Computationally Constrained Grammar Formalisms

- But what if the computational system actually provides constraints on what can and cannot happen?
- The linguist would be forced to assume ad-hoc restrictions to adequately explain the facts
- Better to start with a constrained system and grow only if necessary (Occam's Razor)
- Read: (Gazdar, 1981, Linguistic Inquiry) and Remarks and Replies by E. Willams

Overview

- Tree Adjoining Grammars

$$\text{TAGs}\begin{cases}\text{\sout{Weak vs. Strong Generative Capacity}}\\ \text{\sout{Natural Language and Complexity}}\\ \text{Lexicalization of Grammars}\end{cases}$$

- TAGs for Natural Languages

Lexicalization of Grammars

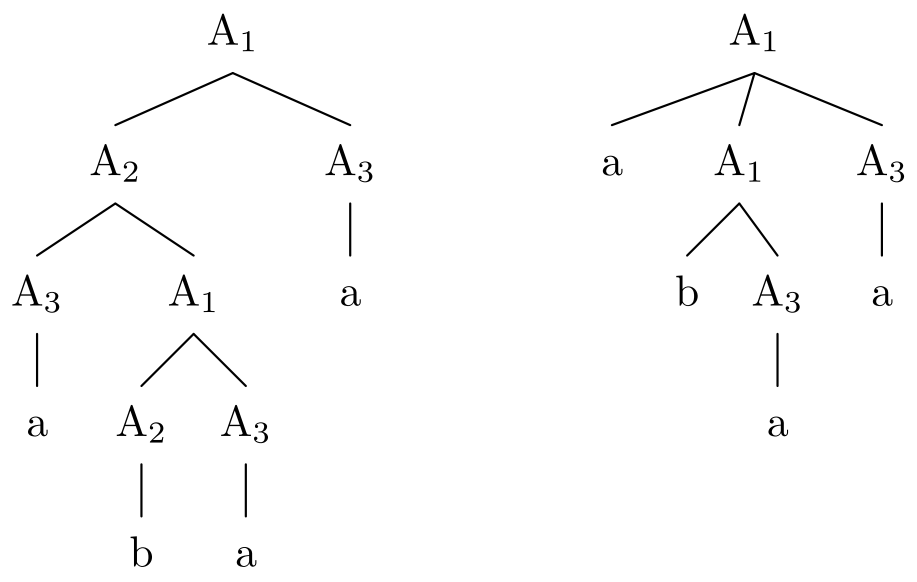
- Why is lexicalization important? Here's a strange fact: a context-free grammar can be *infinitely* ambiguous.
- Lexicalization of a grammar means that each elementary object is associated with some terminal symbol. This guarantees that every input is only *finitely ambiguous*.
  (Joshi and Schabes, 1997) show TALs are *closed* under lexicalization: every TAL has a lexicalized grammar without changing the language.
- Lexicalization is an interesting idea for syntax, semantics (in linguistics) and sentence processing (in psycholinguistics):
  What if each word brings with it an entire syntactic and semantic context for that word?

Lexicalization of Context-Free Grammars

- Can we transform every CFG to a normal form where there is guaranteed to be a terminal symbol on the right hand side?
- Answer: yes – using Griebach Normal Form
- Every CFG can be converted to the form: $A \rightarrow a\ \alpha$ where A is a non-terminal, a is a terminal symbol, and $\alpha \in N^*$

Griebach Normal Form: $T(G) \neq T(GNF(G))$

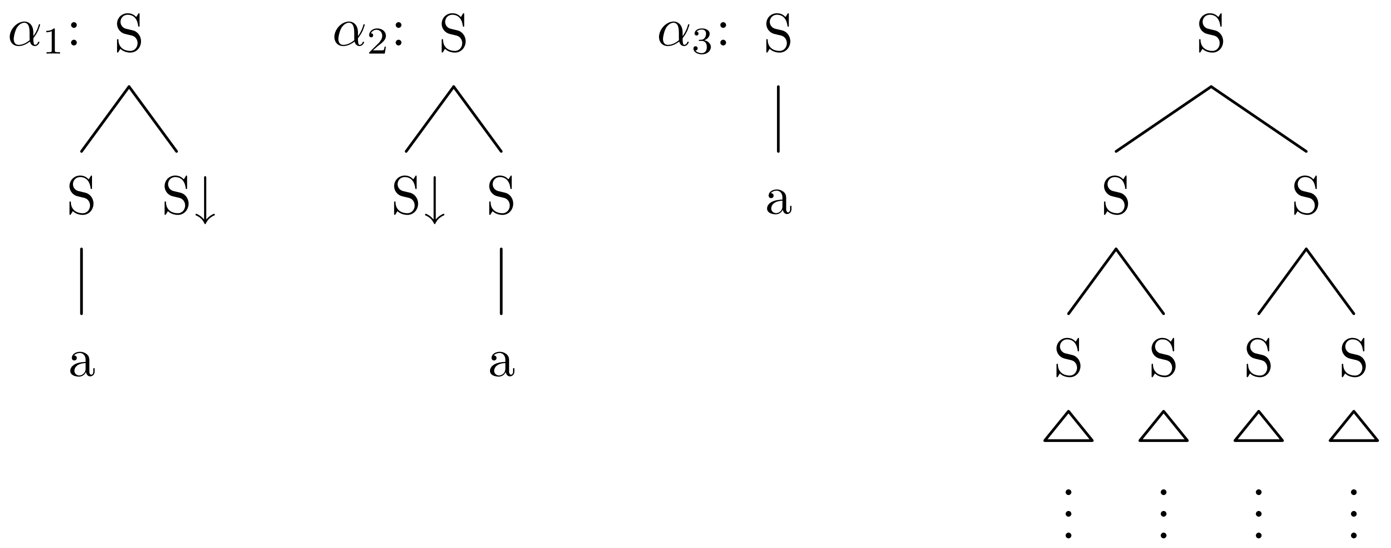
$$\begin{aligned} A_1 &\rightarrow A_2\, A_3 \\ A_2 &\rightarrow A_3\, A_1 \mid b \\ A_3 &\rightarrow A_1\, A_2 \mid a \end{aligned}$$



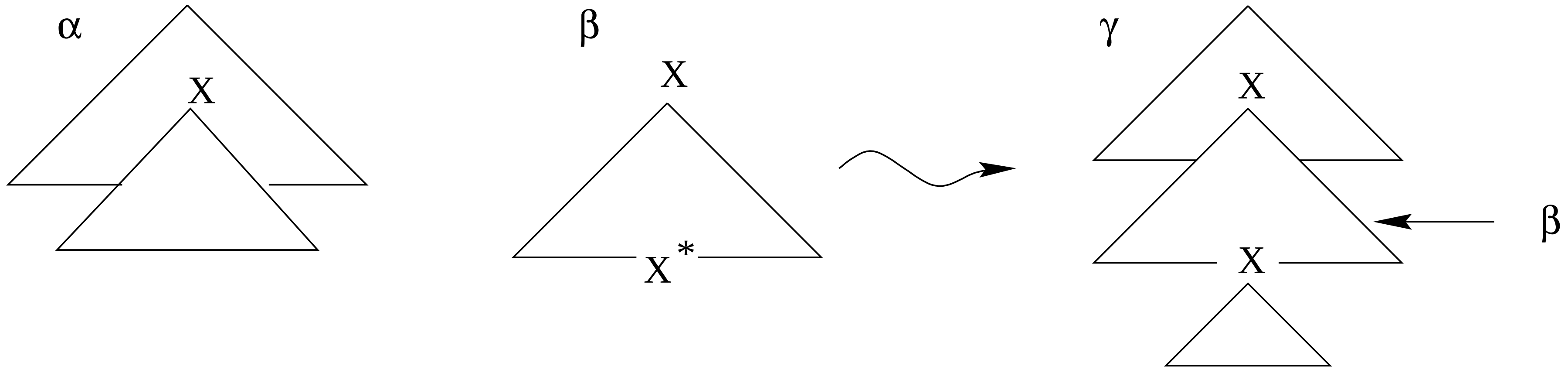
Lexicalization of Context-Free Grammars

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

- Tree substitution Grammar G':



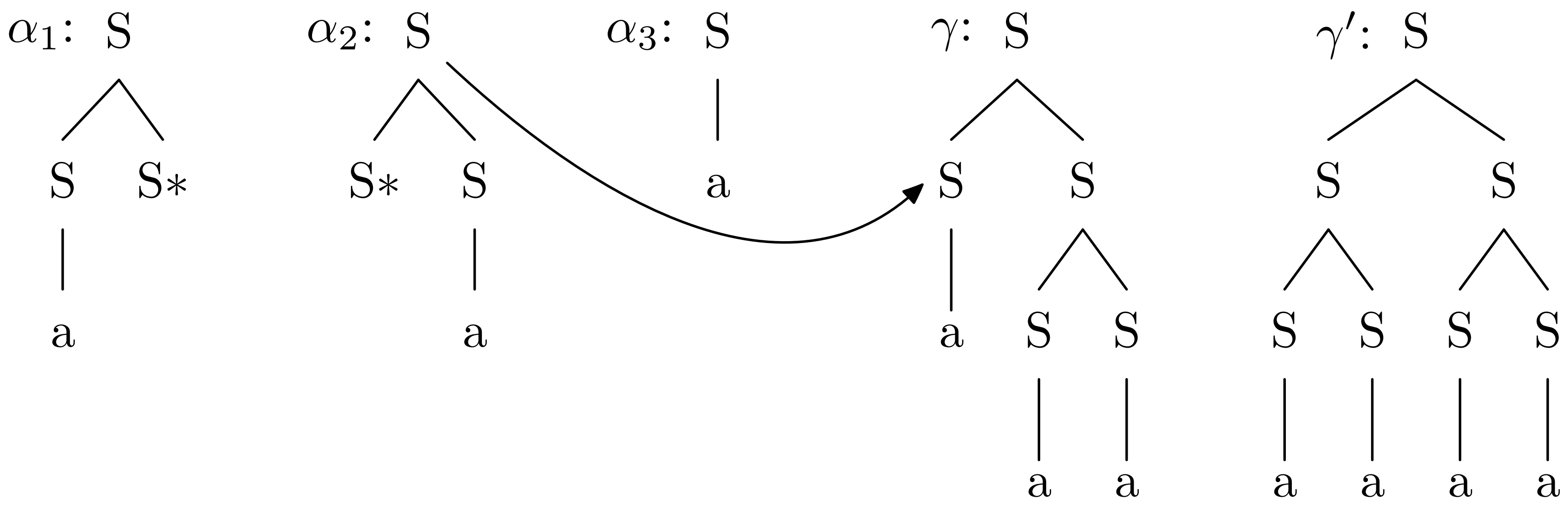
Lexicalization of Context-Free Grammars



Lexicalization of Context-Free Grammars

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

- **Lexicalized** Tree adjoining Grammar G'':



Overview

- Tree Adjoining Grammars

$$\text{TAGs}\begin{cases}\text{~~Weak vs. Strong Generative Capacity~~}\\ \text{~~Natural Language and Complexity~~}\\ \text{~~Lexicalization of Grammars~~}\end{cases}$$

- TAGs for Natural Languages

Lexicalized Tree Adjoining Grammars

- Finite set of elementary trees, such that each tree has at least one terminal symbol
- Elementary trees: initial and auxiliary
- Operations: substitution and adjunction
- Derivation Tree: how elementary trees are put together
- Derived Tree

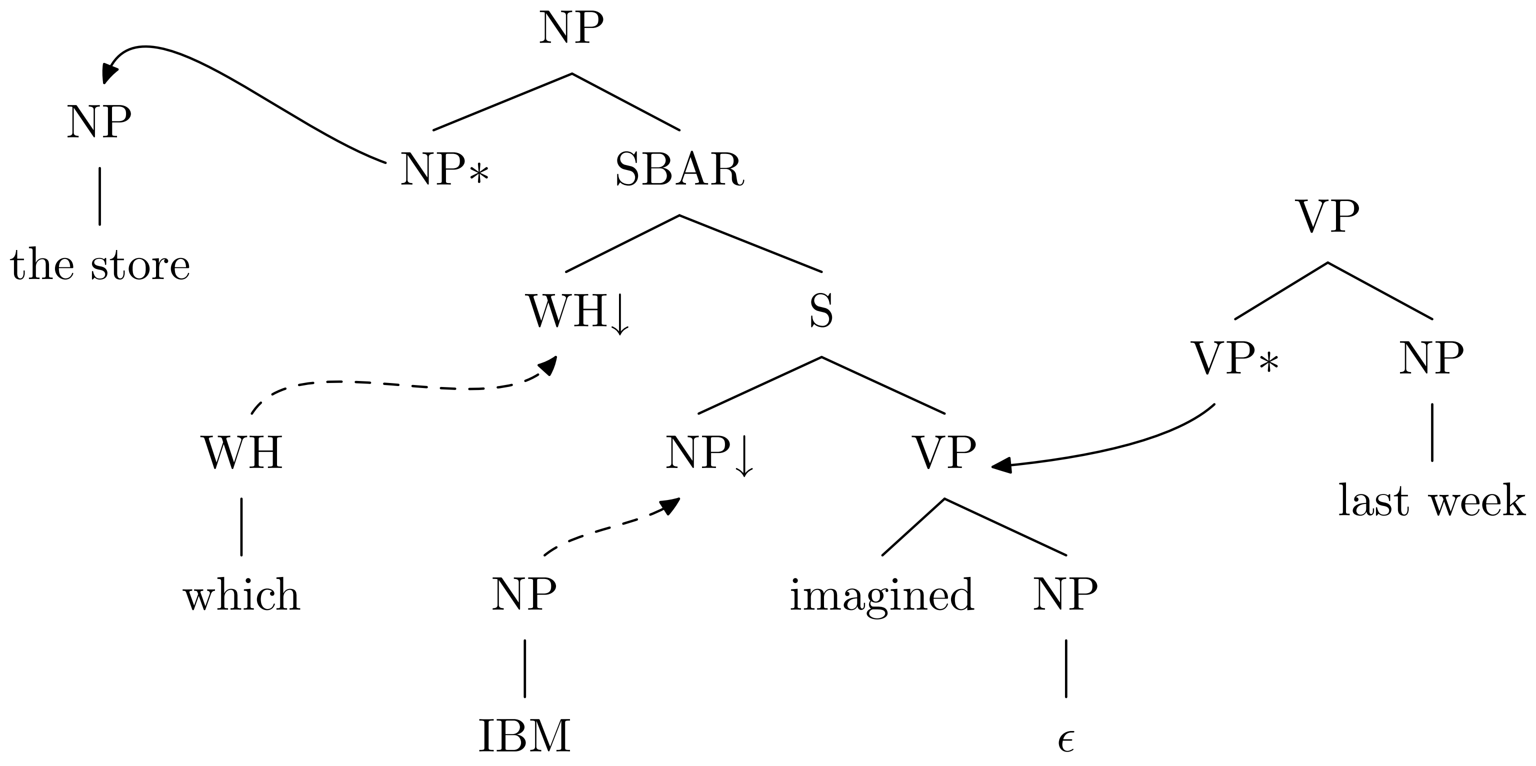
Localization of Dependencies

- Syntactic:
  - agreement: person, number, gender
  - subcategorization: sleeps (null), eats (NP); gives (NP NP)
  - filler-gap: who did John ask Bill to invite ϵ
  - word order: within and across clauses as in scrambling, clitic movement, etc.

Localization of Dependencies

- Semantic:
  - function-argument: all arguments of the anchoring word (the *functor*) are localized
  - word clusters (flexible idioms): non-compositional, e.g. take a walk, give a cold shoulder to
  - word co-occurences, lexical semantic aspects of meaning

Lexicalized Tree Adjoining Grammars


NP
the store
NP
NP∗
SBAR
WH↓
S
WH
which
NP↓
VP
NP
IBM
imagined
NP
ϵ
VP
VP∗
NP
last week