(Just a bit) beyond context-free

Computational Linguistics (LING 455)

Rutgers University December 3, 2021

Kinds of rewrite rules

CFGs have rewrite rules of the shape $A \longrightarrow \varphi$, where φ is a string of terminals and non-terminals.

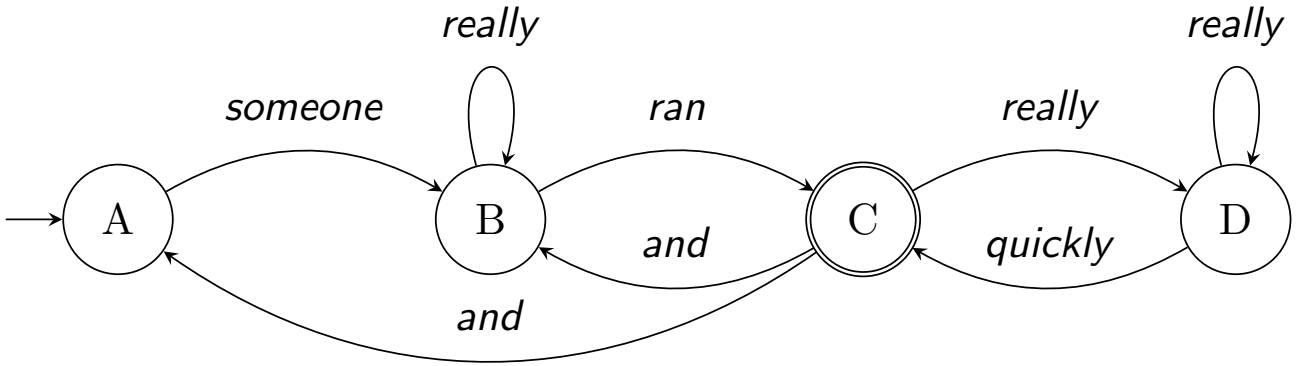
- The restriction is on the LHS. This is what makes it context-free.

We can contemplate restrictions on the RHS as well:

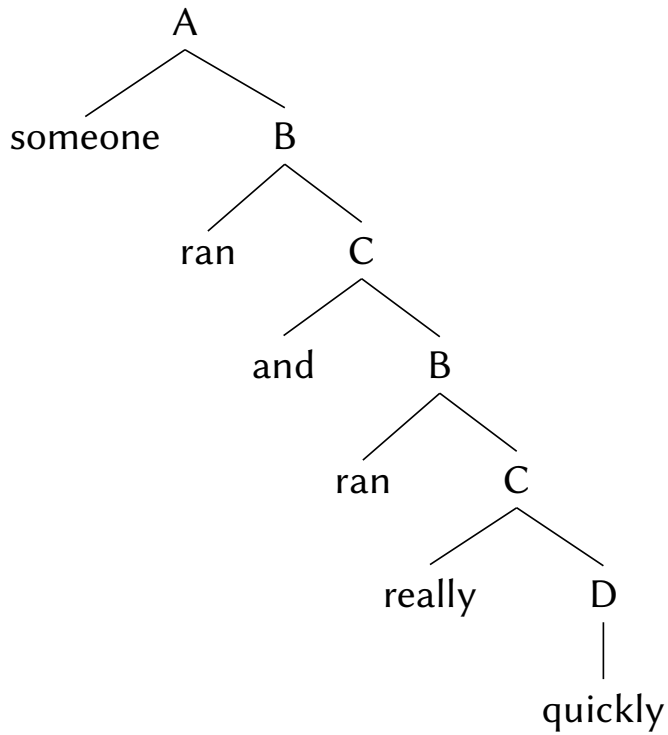
- Rules like $A \longrightarrow x$, $A \longrightarrow xB$ only: finite-state/regular
- Rules like $A \longrightarrow x$, $A \longrightarrow BC$ only: Chomsky Normal Form

CNF is a convenience, not a restriction in expressive power.

FSAs as rewrite systems (cont)



A → someone B
B → really B
B → ran C
B → ran
C → and A
C → and B
C → really D
D → really D
D → quickly C
D → quickly



Center embedding

The book stinks.

Center embedding

The book stinks.
DP VP

Center embedding

The book	this guy	wrote	stinks.
DP	DP	VP	VP

Center embedding

The book	this guy	I	know	wrote	stinks.
DP	DP	DP	VP	VP	VP

Center embedding

The book	this guy	I	know	wrote	stinks.
DP	DP	DP	VP	VP	VP

*

The book	this guy	I	know		stinks.
DP	DP	DP	VP		VP

Center embedding is non-finite-state

There are a few ways to see this. First, we can recall the pumping lemma for regular lgs, which applies to FSAs just as well:

- FSAs for infinite lgs must have a loop. This means that (like REs) any (sufficiently long) FSA string must be pumpable.
- Pick an $a^n b^n$ for any $n \geqslant 1$. No part of it can be pumped without either (i) creating unbalanced numbers of a's and b's, or (ii) creating an aba sequence. Both illegal.

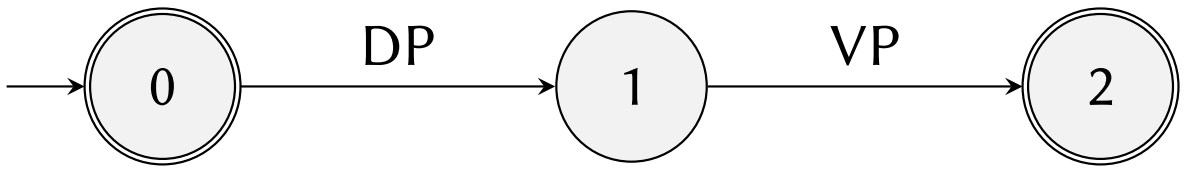
Center embedding is non-finite-state

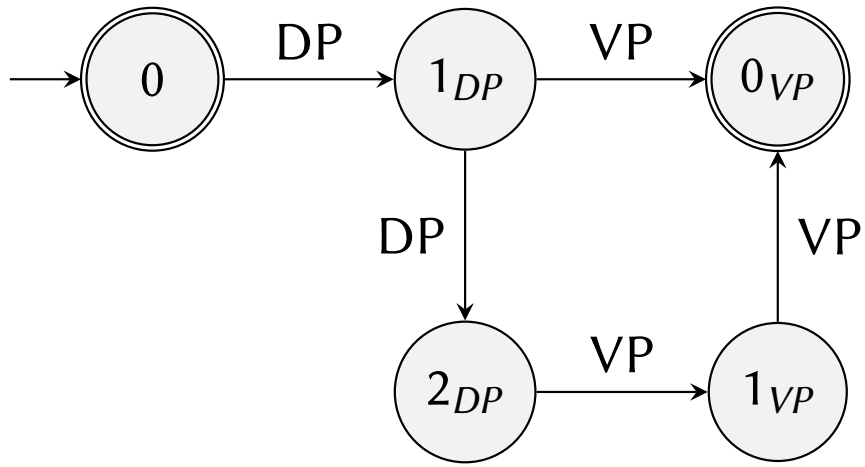
There are a few ways to see this. First, we can recall the pumping lemma for regular lgs, which applies to FSAs just as well:

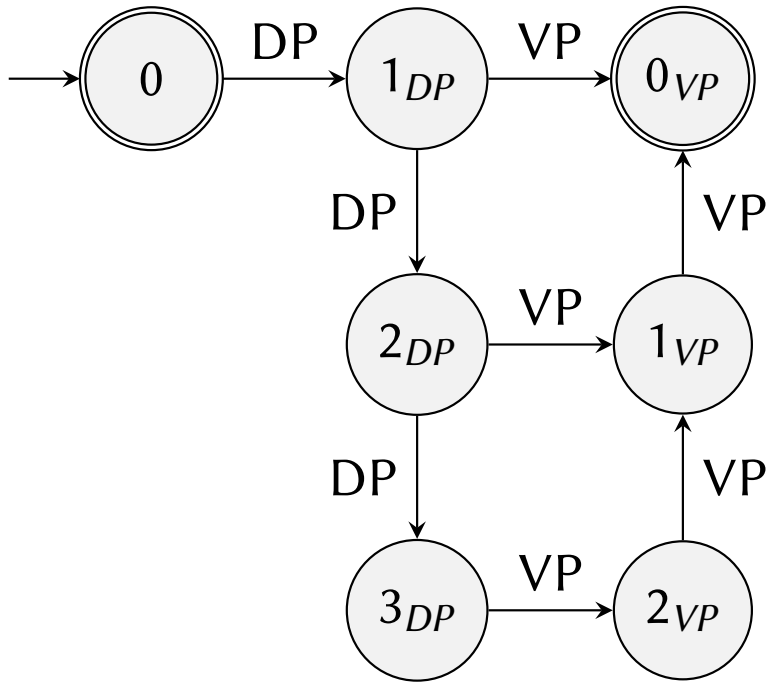
- FSAs for infinite lgs must have a loop. This means that (like REs) any (sufficiently long) FSA string must be pumpable.
- Pick an $a^n b^n$ for any $n \geqslant 1$. No part of it can be pumped without either (i) creating unbalanced numbers of *a*'s and *b*'s, or (ii) creating an *aba* sequence. Both illegal.

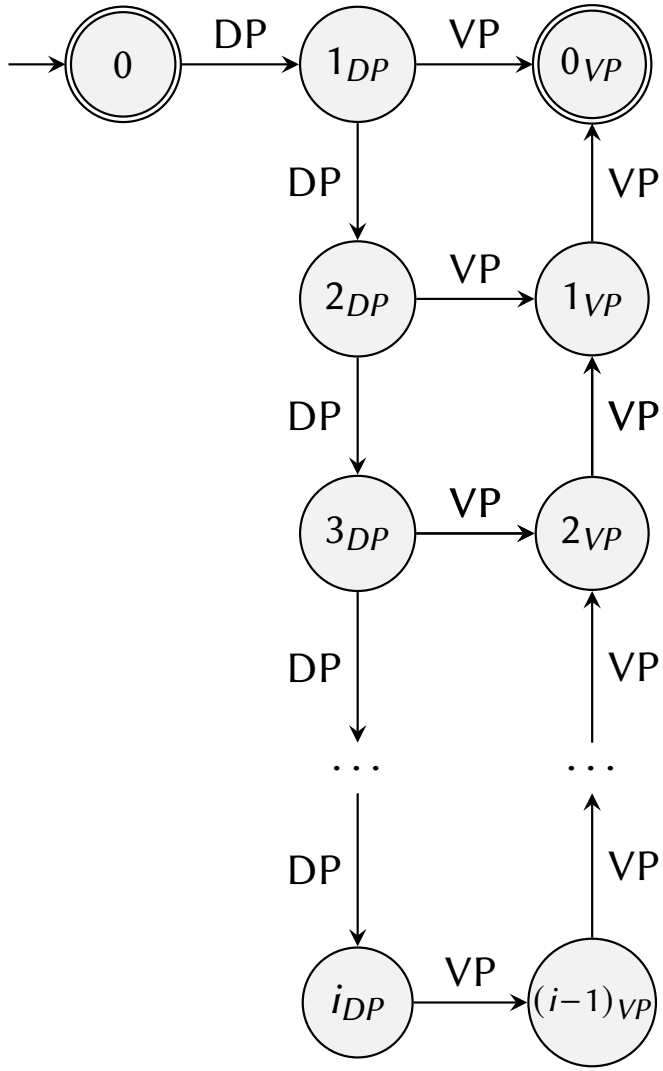
Or: the forward set for a string tells us "what we can do next".

- Because finite state machines have finite #s of states, there can only be finitely many forward-set "buckets" (Myhill-Nerode).
- Yet each string of *a*'s has a different "what you can do next"!

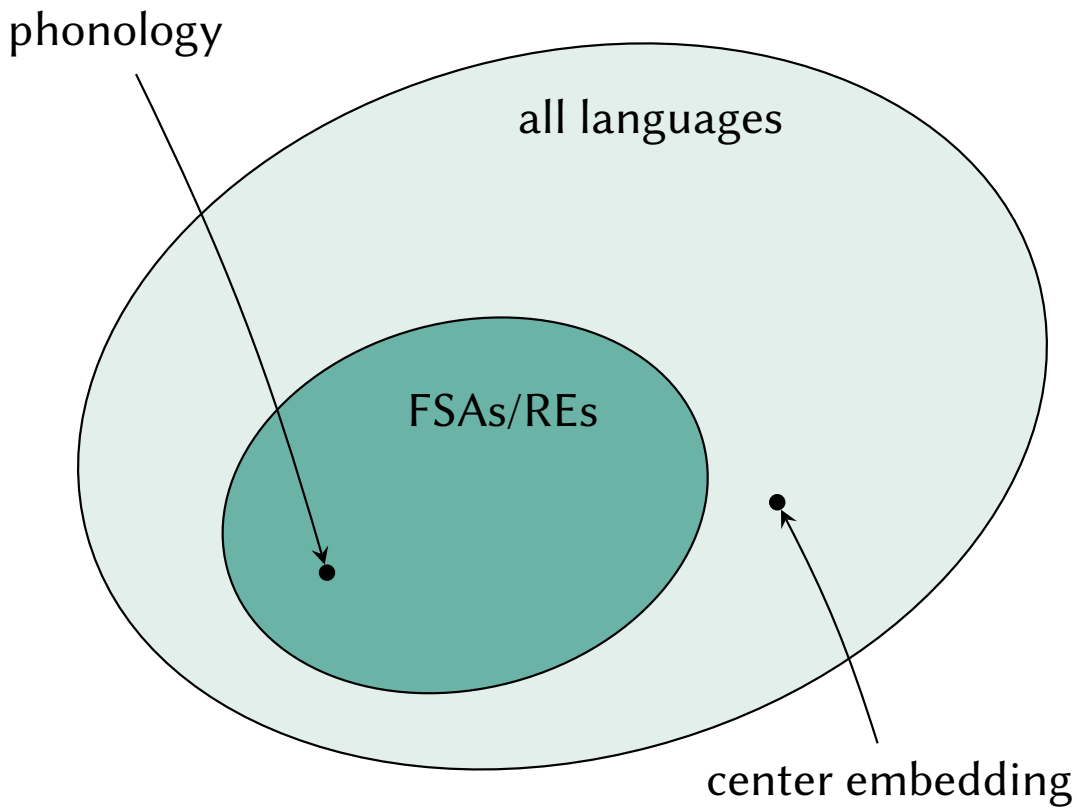

0
DP
1
VP
2


0
DP
1_{DP}
VP
0_{VP}
DP
VP
2_{DP}
VP
1_{VP}

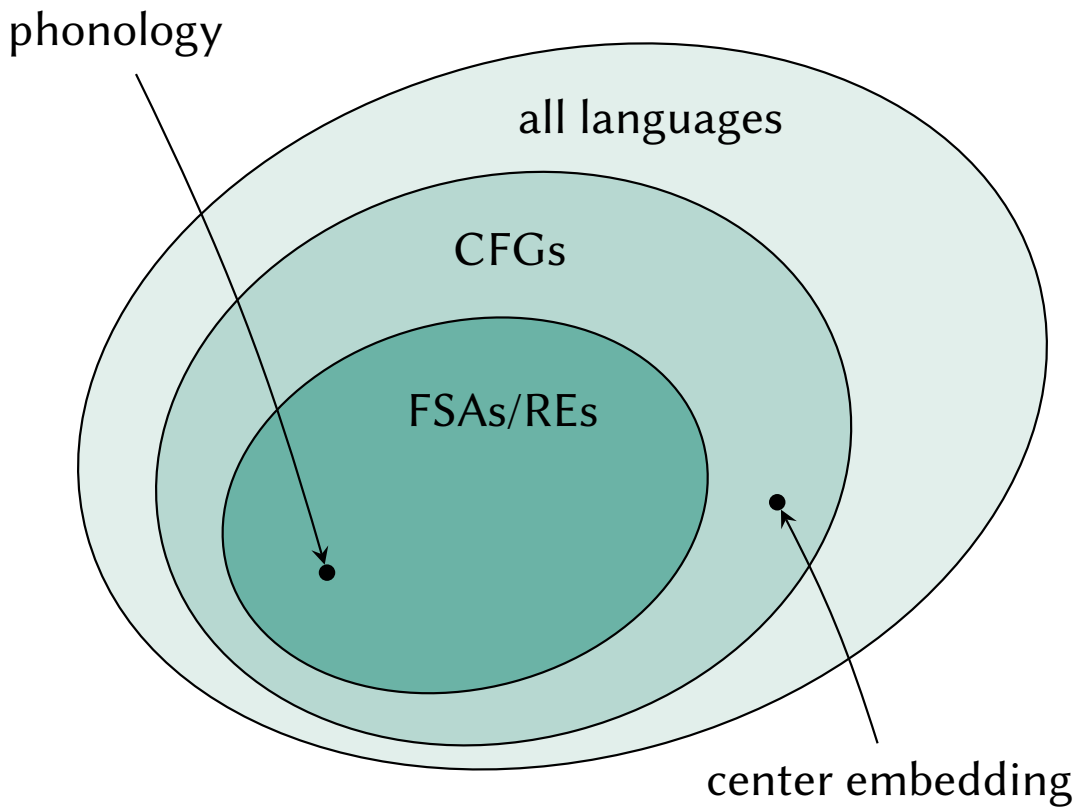

0
DP
1_{DP}
VP
0_{VP}
DP
VP
2_{DP}
VP
1_{VP}
DP
VP
3_{DP}
VP
2_{VP}


0
DP
1_{DP}
VP
0_{VP}
DP
VP
2_{DP}
VP
1_{VP}
DP
VP
3_{DP}
VP
2_{VP}
DP
VP
...
...
DP
VP
i_{DP}
VP
$(i-1)_{VP}$

Hierarchy of language types



Hierarchy of language types



Movement

These are reasonably strong motivations for considering natural lgs to require at least context-free power to characterize.

- But is this *enough* power?

Movement

These are reasonably strong motivations for considering natural lgs to require at least context-free power to characterize.

- But is this *enough* power?

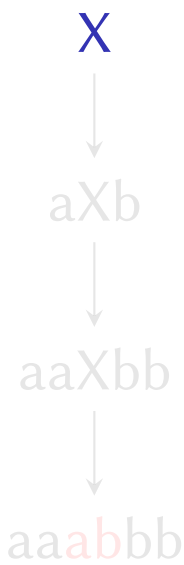
It was recognized early on (in the classic, pioneering papers by Chomsky) that **displacement** creates some challenges for CFGs:

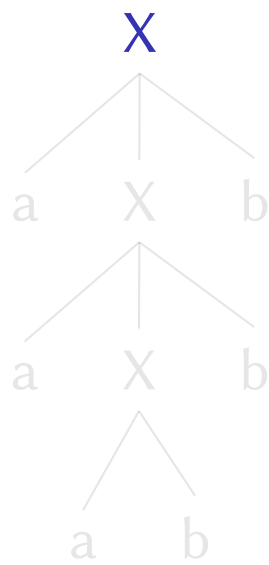
(1) This is the book I read _.

(2) This is the longest book I thought Tolstoy wrote _.

In CFG terms, *book* is clearly in some real sense the **object** of *read* and of *wrote*. Yet it is pronounced far away (in English).

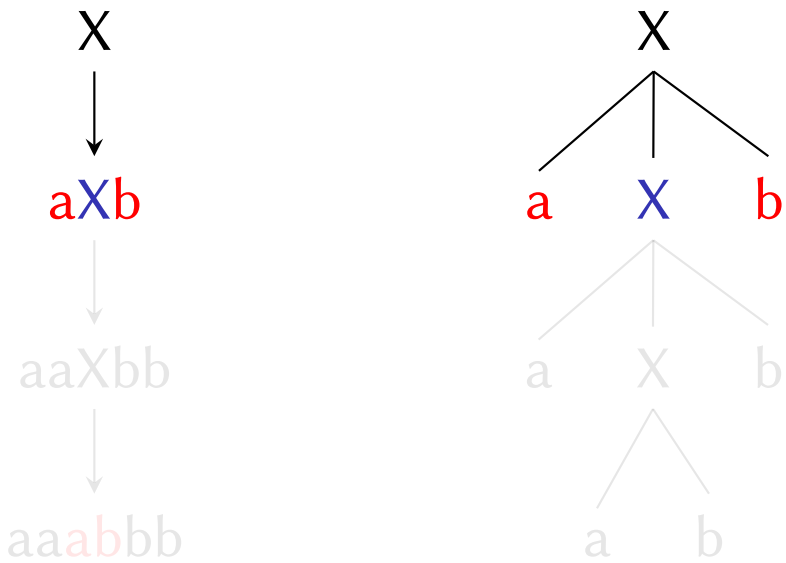
$X \to ab; X \to aXb$





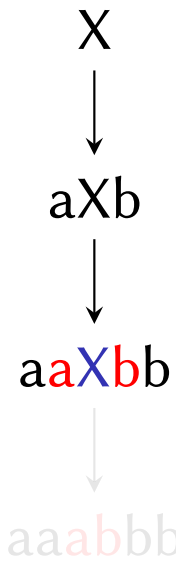
Σ^+ is the non-empty strings in Σ^*.

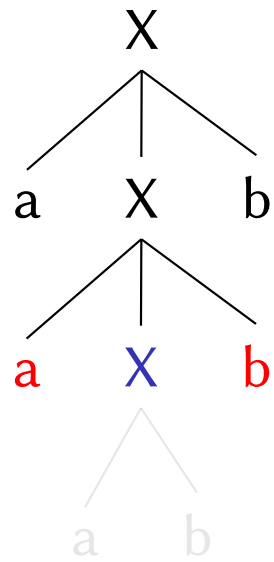
$X \longrightarrow ab; X \longrightarrow aXb$



Σ^+ is the non-empty strings in Σ^*.

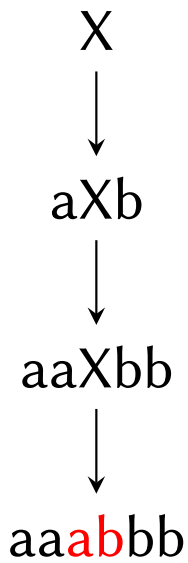
$X \longrightarrow ab; X \longrightarrow aXb$

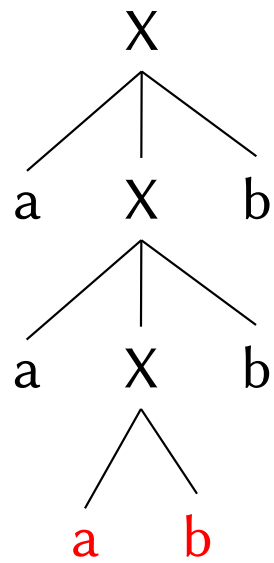




Σ^+ is the non-empty strings in Σ^*.

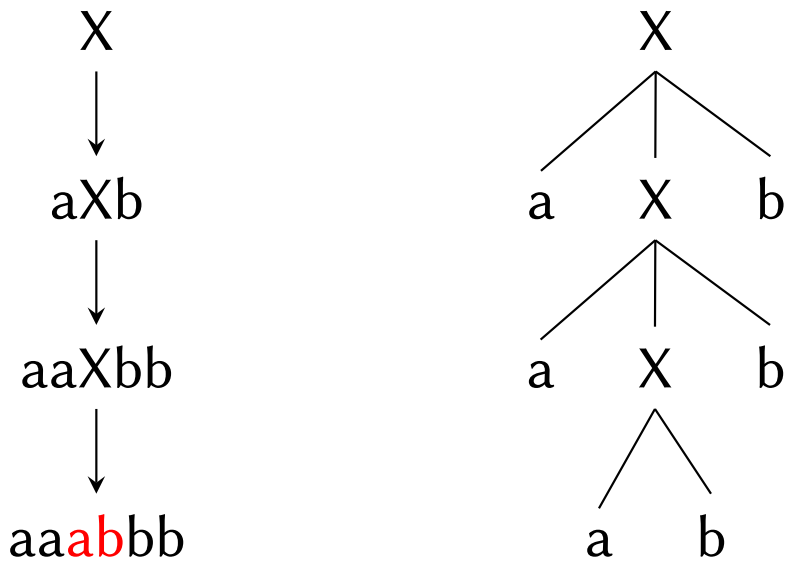
$X \longrightarrow ab; X \longrightarrow aXb$





Σ^+ is the non-empty strings in Σ^*.

$X \to ab; X \to aXb$



The CFG generates aaabbb as aaabbb, **not** aaabbb. Relatedly:

- CFGs **can** generate palindrome lgs, $\{ww^R \mid w \in \{a, b\}^+\}$. (How?)
- But CFGs **cannot** generate repeat lgs: $\{ww \mid w \in \{a, b\}^+\}$!

Σ^+ is the non-empty strings in Σ^*.

Cross-serial dependencies in natural lgs (Shieber 1985)

CFGs easily generated **nested** dependencies, but not **crossed** ones.
Is there evidence for crossed dependencies in human lgs?

Cross-serial dependendencies in natural lgs (Shieber 1985)

CFGs easily generated **nested** dependencies, but not **crossed** ones.
Is there evidence for crossed dependencies in human lgs?

There is! The famous examples are from Swiss German (similar if less conclusive patterns observed in High German, Dutch):

(3) …das mer d'chind em Hans es huus lönd hälfe aastriiche
…that we the kids-acc Hans-dat house-acc let help paint
'…that we let the kids help Hans paint the house'

Abstractly, the pattern looks like this:

- CN_aV_a
- CN_dV_d
- $CN_aN_aN_dV_aV_aV_d$
- …

Other examples (Stabler 2004)

Mandarin 'A-not-A' polar questions:

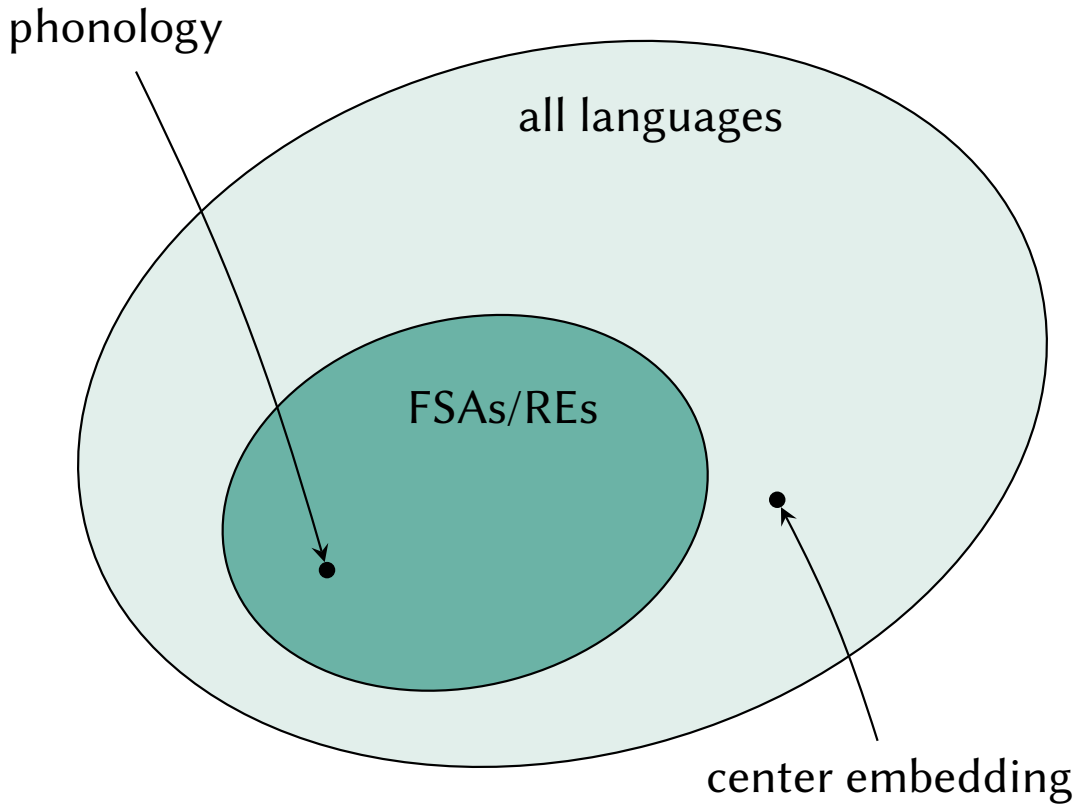
(4) Zhangsan ai da lanqiu, bu ai da lanqiu?

Phrasal ellipsis (English and elsewhere):

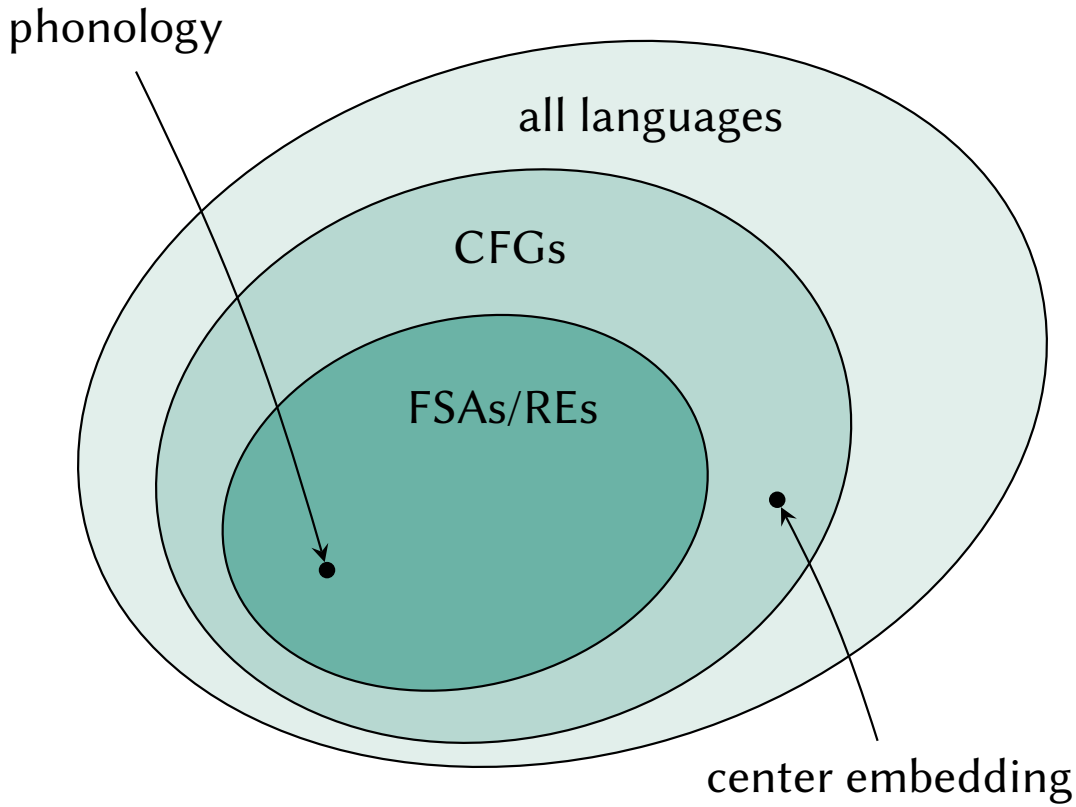
(5) I gave Mary a present before you could (give Mary a present).

The analyses of these constructions are not uncontroversial, but their resemblance to cross-serial dependencies is clear.

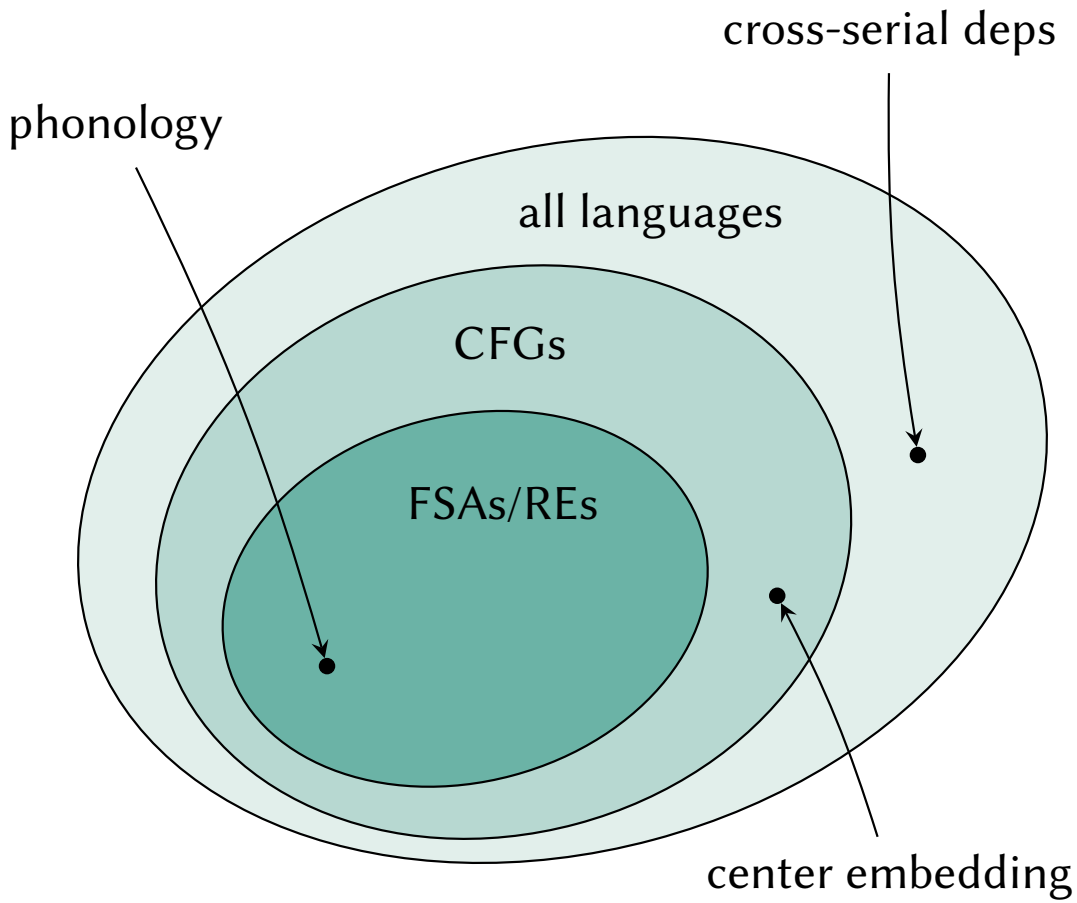
So is natural lg context-sensitive after all?



So is natural lg context-sensitive after all?



So is natural lg context-sensitive after all?



No way!

CSG's are far **too expressive** to represent a plausible model of linguistic (syntactic) competence. Here's a CSG lg:

- $\{a^n \mid n \text{ is a prime number}\}$

CSG's are also **intractable** to parse.

- Parsing for CFG's can be made tractable/polynomial, $O(n^3)$
- Parsing CSG's is exponential, $O(x^n)$

A switch in perspective and notation

A CFG rule $A \longrightarrow BC$ can be read top-down or bottom-up:

- Top-down: A is rewritten as ('goes to') BC
- Bottom-up: if I have a B followed by a C, I know I have an A

We can adopt a bottom-up notation that refers more directly to the strings related by the relevant rule:

$$A(uv) \leftarrow B(u)\ C(v)$$

'If u is a B, and v is a C, then uv is an A'. Some examples:

- $\mathrm{S}(uv) \leftarrow \mathrm{DP}(u)\ \mathrm{VP}(v)$
- $\mathrm{NP}(\texttt{elk})$, $\mathrm{P}(\texttt{with})$, ...

Pairs in Multiple CFGs

With this shift in notation, we can adopt rules that categorize and build **pairs of strings**:

- $\mathrm{Y}(\mathrm{a}, \mathrm{b})$
- $\mathrm{Y}(\mathrm{a}u, \mathrm{b}v) \leftarrow \mathrm{Y}(u, v)$
- $\mathrm{X}(uv) \leftarrow \mathrm{Y}(u, v)$

$\mathrm{Y}(\mathrm{a}, \mathrm{b})$

↓

$\mathrm{Y}(\mathrm{aa}, \mathrm{bb})$

↓

$\mathrm{Y}(\mathrm{aaa}, \mathrm{bbb})$

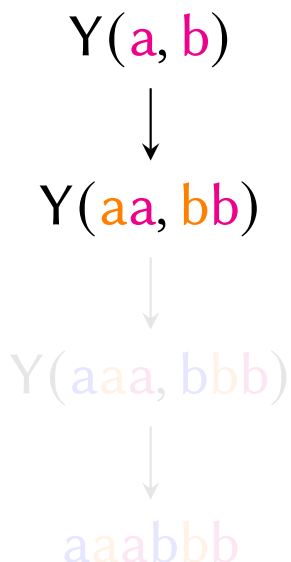
↓

aaabbb

Pairs in Multiple CFGs

With this shift in notation, we can adopt rules that categorize and build **pairs of strings**:

- $\mathrm{Y}(\mathrm{a}, \mathrm{b})$
- $\mathrm{Y}(\mathrm{a}u, \mathrm{b}v) \leftarrow \mathrm{Y}(u, v)$
- $\mathrm{X}(uv) \leftarrow \mathrm{Y}(u, v)$



Pairs in Multiple CFGs

With this shift in notation, we can adopt rules that categorize and build **pairs of strings**:

- $\mathrm{Y}(\mathrm{a}, \mathrm{b})$
- $\mathrm{Y}(\mathrm{a}u, \mathrm{b}v) \leftarrow \mathrm{Y}(u, v)$
- $\mathrm{X}(uv) \leftarrow \mathrm{Y}(u, v)$

$$\mathrm{Y}(\mathrm{a}, \mathrm{b})$$
$$\downarrow$$
$$\mathrm{Y}(\mathrm{aa}, \mathrm{bb})$$
$$\downarrow$$
$$\mathrm{Y}(\mathrm{aaa}, \mathrm{bbb})$$
$$\downarrow$$
$$\mathrm{aaabbb}$$

Pairs in Multiple CFGs

With this shift in notation, we can adopt rules that categorize and build **pairs of strings**:

- $\mathrm{Y}(\mathrm{a}, \mathrm{b})$
- $\mathrm{Y}(\mathrm{a}u, \mathrm{b}v) \leftarrow \mathrm{Y}(u, v)$
- $\mathrm{X}(uv) \leftarrow \mathrm{Y}(u, v)$

$$\mathrm{Y}(\mathrm{a}, \mathrm{b})$$
$$\downarrow$$
$$\mathrm{Y}(\mathrm{aa}, \mathrm{bb})$$
$$\downarrow$$
$$\mathrm{Y}(\mathrm{aaa}, \mathrm{bbb})$$
$$\downarrow$$
$$\mathrm{aaabbb}$$

Exactly what we need

This is exactly what we need to generate cross-serial dependencies in (e.g.) Swiss German, and wherever else they are found.

- $\mathrm{T}(\mathrm{N}_a, \mathrm{V}_a)$
- $\mathrm{T}(\mathrm{N}_d, \mathrm{V}_d)$
- $\mathrm{T}(\mathrm{N}_a u, \mathrm{V}_a v) \leftarrow \mathrm{T}(u, v)$
- $\mathrm{T}(\mathrm{N}_d u, \mathrm{V}_d v) \leftarrow \mathrm{T}(u, v)$
- $\mathrm{S}(\mathrm{C}uv) \leftarrow \mathrm{T}(u, v)$

With these rules, easy to generate $\mathrm{C}\mathrm{N}_a\mathrm{N}_a\mathrm{N}_d\mathrm{V}_a\mathrm{V}_a\mathrm{V}_d$.

Movement

Pairs represent **discontinuous constituents**, constituents with a bit of extra structure allowing us to identify a break point.

This affords a natural treatment of movement:

- $\mathrm{NVP}(u, v) \leftarrow \mathrm{NP}(u)\ \mathrm{TV}(v)$ (note: RHS order isn't meaningful)
- $\mathrm{NS}(u, wv) \leftarrow \mathrm{DP}(w)\ \mathrm{NVP}(u, v)$
- $\mathrm{NRP}(u, wv) \leftarrow \mathrm{R}(w)\ \mathrm{NS}(u, v)$
- $\mathrm{NP}(uv) \leftarrow \mathrm{NRP}(u, v)$

MCFGs

…Are much more restrictive and much more natural as models of linguistic (syntactic) competence than CSGs.

…Have a tractable parsing problem. Grammars of the shape we considered today can be parsed in polynomial time, $O(n^6)$.