Natural Language Processing

Lecture 6: Introduction to Syntax and Formal Languages.

09/20/2024

COMS W4705
Daniel Bauer

Sentences: the good, the bad, and the ugly

- Some good sentences:
  - *the boy likes a girl*
  - *the small girl likes a big girl*
  - *a very small nice boy sees a very nice boy*
- Some bad sentences:
  - *the boy the girl likes*
  - *small boy likes nice girl*
- Ugly word salad: *very like nice the girl boy*

Syntax

- Syntax is the study of **structure** of language
  - How words are arranged in a sentence (word order) and the relationship between them.
  - Goal: relate surface form to semantics (meaning).

Syntax as an Interface

- Syntax can be seen as the interface between morphology (structure of words) and semantics.
- Why treat syntax separately from semantics?
    - Can judge if a sentence is grammatical or not, even if it doesn't make sense semantically.

Colorless green ideas sleep furiously.

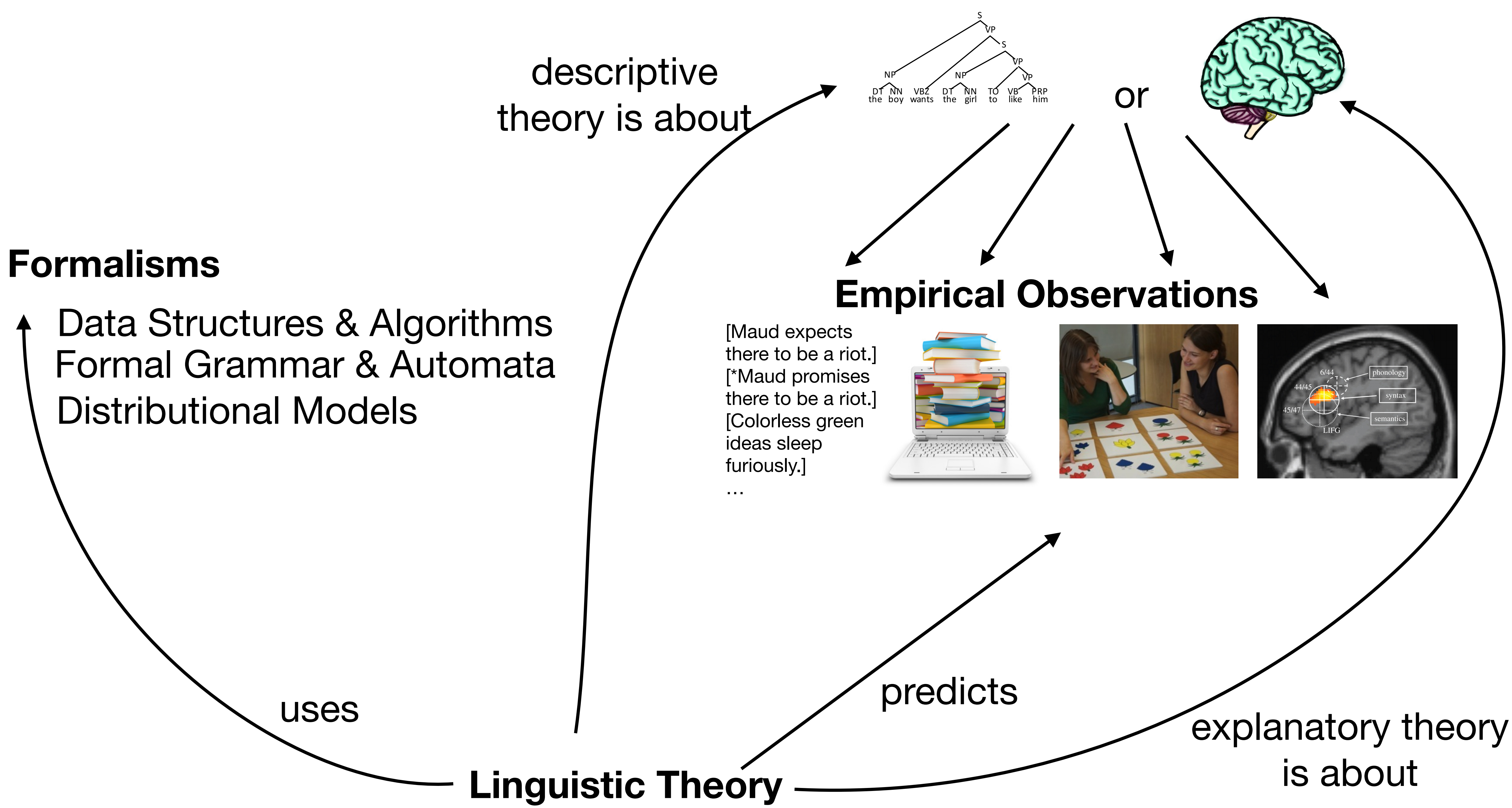
**Sleep ideas furiously colorless green.*

Types of Linguistic Theories

- **Prescriptive:** “This is how people ought to talk.”
  - (“prescriptive linguistics” is an oxymoron)
- **Descriptive:** provide a formal account of **how** people talk.
- **Explanatory:** explain **why** people talk a certain way (identify underlying cognitive, or neural mechanism)

NLP focuses on the descriptive part.
Computational linguistics is interested in finding explanatory theories, but often uses descriptive methods.

The Big Picture



Constituents

- A constituent is a group of words that behave as a single unit (within a hierarchical structure).

- Noun-Phrase examples:

  - [they], [the woman], [three parties from Brooklyn],
    [a high-class spot such as Mindy's], [the horse raced past the barn]

  - Noun phrases can appear before verbs (among other things) and they must be complete:

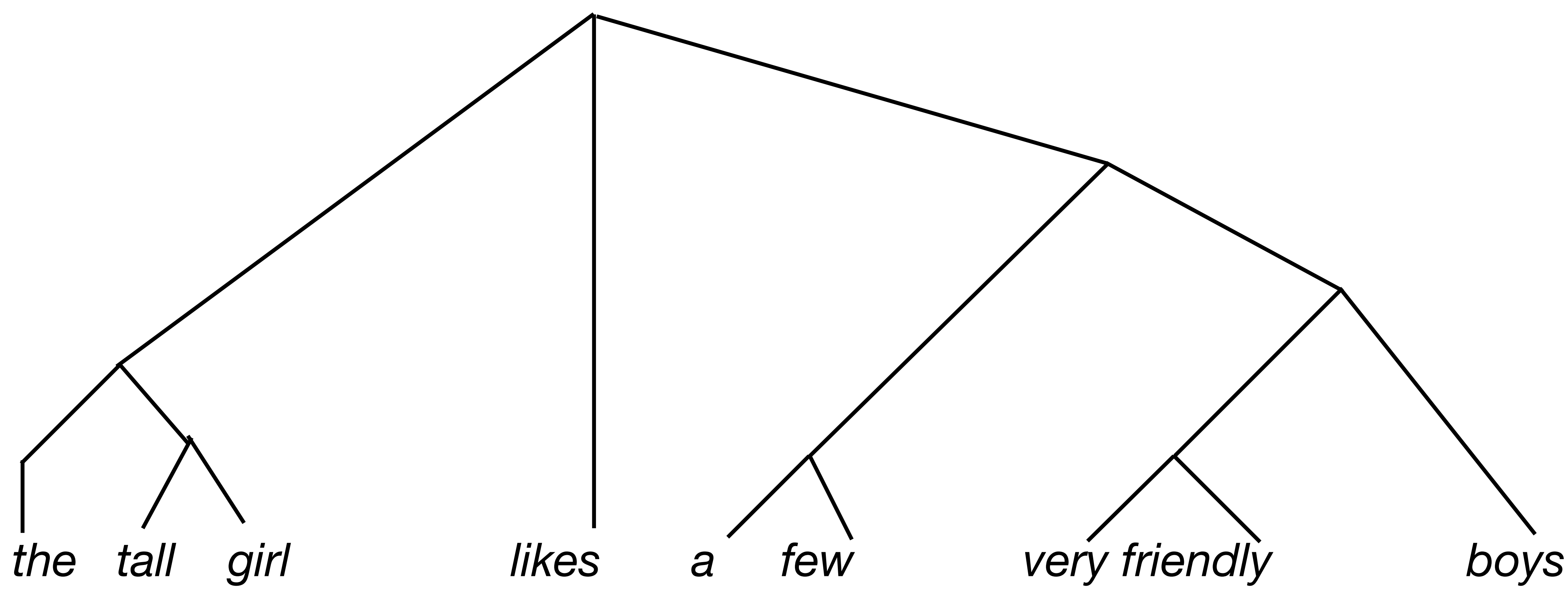
    - **from* ***arrive****…*
      the* ***is* *….*
      spot* ***sat**….*
      green* ***sleep…*

Constituency Tests

- **_On September seventeenth_**_, I’d like to fly to New York._
- _I’d like to fly to New York_ **_on September seventeenth._**
- _I’d like to fly_ **_on September seventeenth_** _to New York._
- _*_**_On_** _I’d like to fly to New York_ **_September seventeeth_**_._
- _*_**_On September_** _I’d like to fly_ **_seventeenth_** _to New York._

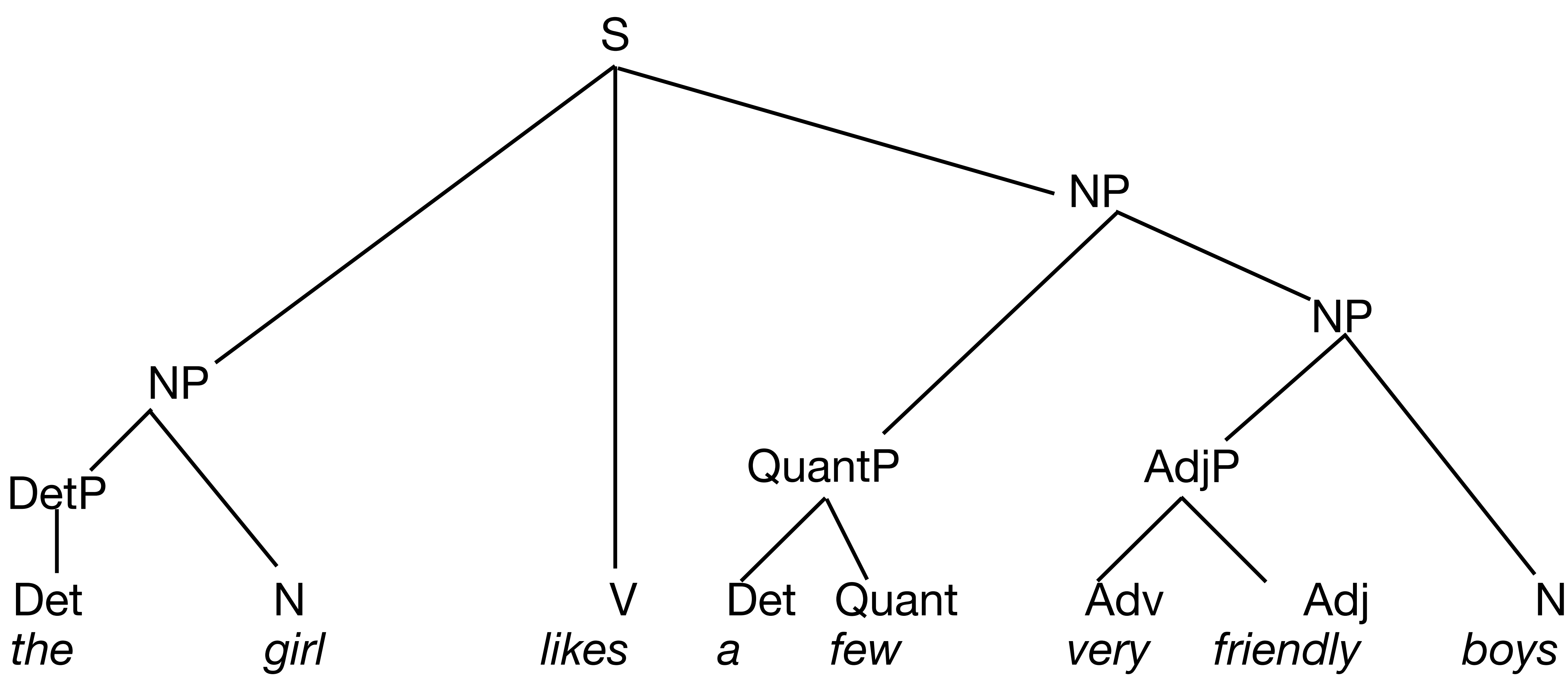
Sentence Structure as Trees

- [the tall girl likes a few very friendly boys]
- [[the tall girl] likes [a few very friendly boys]]
- [[the] tall girl] likes [[a few] [[very friendly] boys]]]



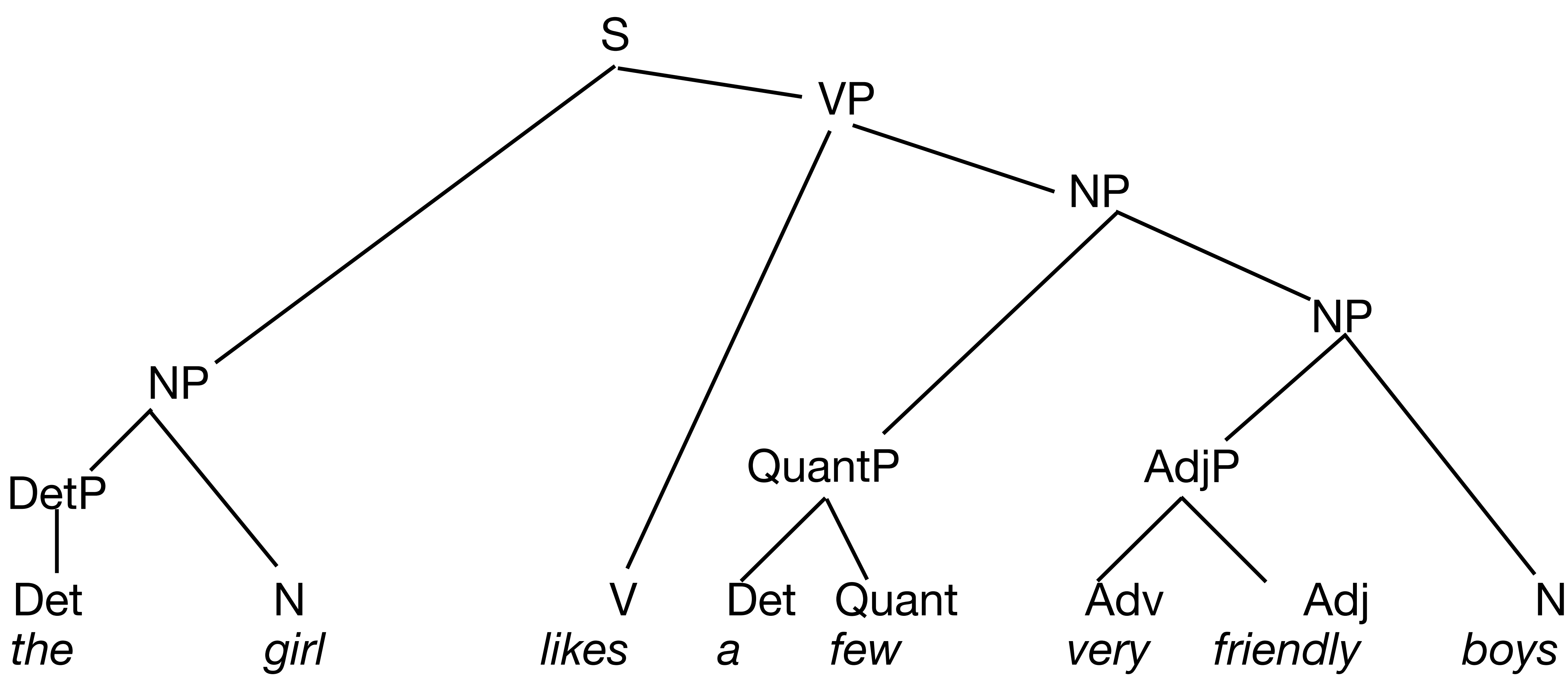
Constituent Labels

- Choose constituents so each one has one non-bracketed word: the **head.**
- Category of Constituent: XP, where X is the part-of-speech of the head
  NP, VP, AdjP, AdvP, DetP



Constituent Labels

- Choose constituents so each one has one non-bracketed word: the **head.**
- Category of Constituent: XP, where X is the part-of-speech of the head
  NP, VP, AdjP, AdvP, DetP



Recursion in Language

- One of the most important attributes of Natural Languages is that they are **recursive.**

  - *He made pie*
    [with apples [from the orchard [near the farm [in …]]]]

  - *[The mouse [the cat [the dog chased]] ate] died.*

- There are infinitely many sentences in a language, but in predictable structures.

- How do we model the set of sentences in a language and their structure?

Context Free Grammars (CFG)

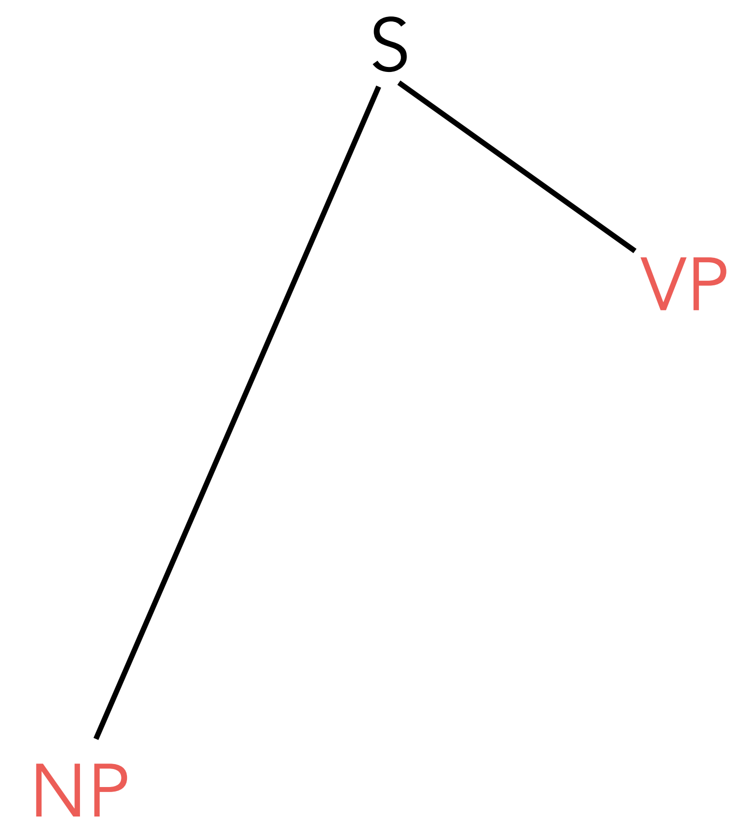
S → NP VP	V → saw
VP → V NP	P → with
VP → VP PP	D → the
PP → P NP	N → cat
NP → D N	N → tail
NP → NP PP	N → student

S

Context Free Grammars (CFG)

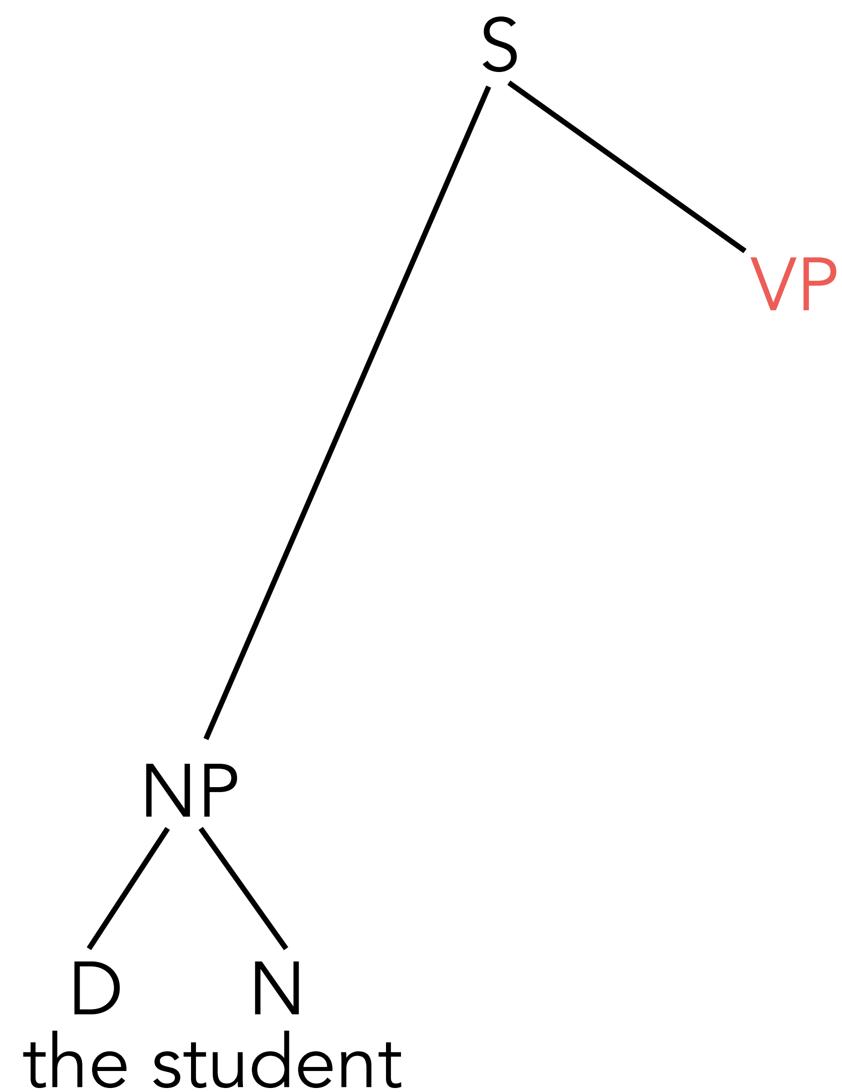
S → NP VP
VP → V NP
VP → VP PP
PP → P NP
NP → D N
NP → NP PP

V → saw
P → with
D → the
N → cat
N → tail
N → student



Context Free Grammars (CFG)

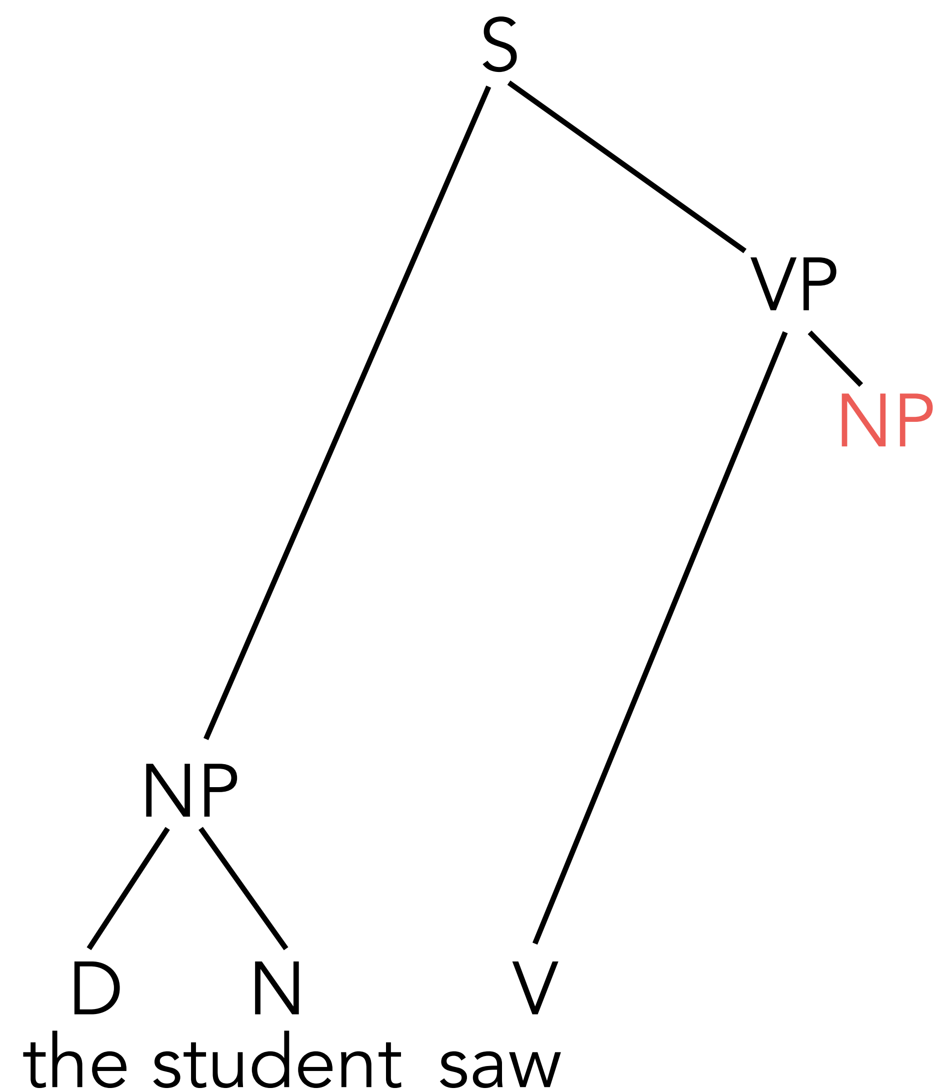
S → NP VP	V → saw
VP → V NP	P → with
VP → VP PP	D → the
PP → P NP	N → cat
NP → D N	N → tail
NP → NP PP	N → student



Context Free Grammars (CFG)

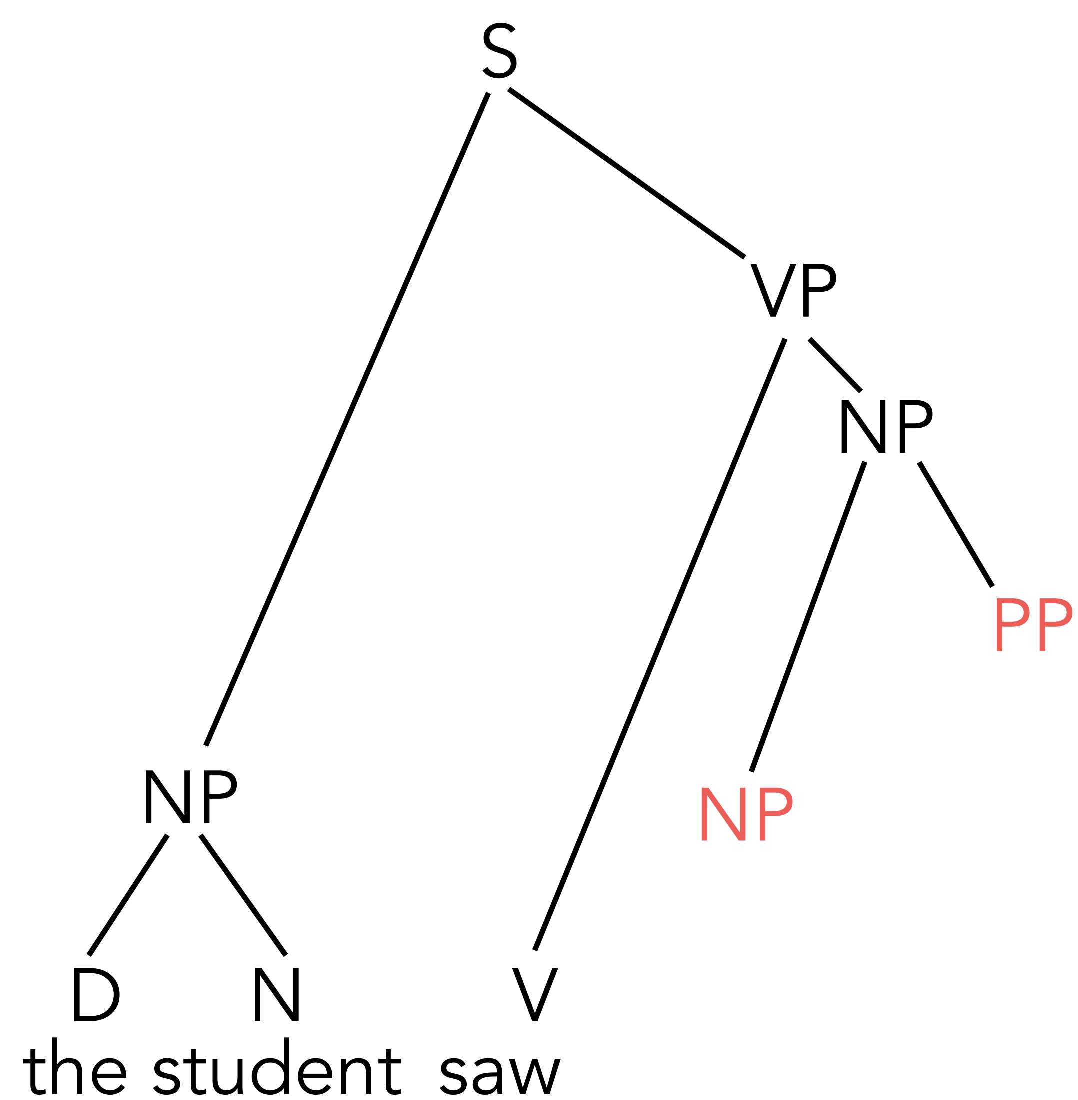
S → NP VP
VP → V NP
VP → VP PP
PP → P NP
NP → D N
NP → NP PP

V → saw
P → with
D → the
N → cat
N → tail
N → student



Context Free Grammars (CFG)

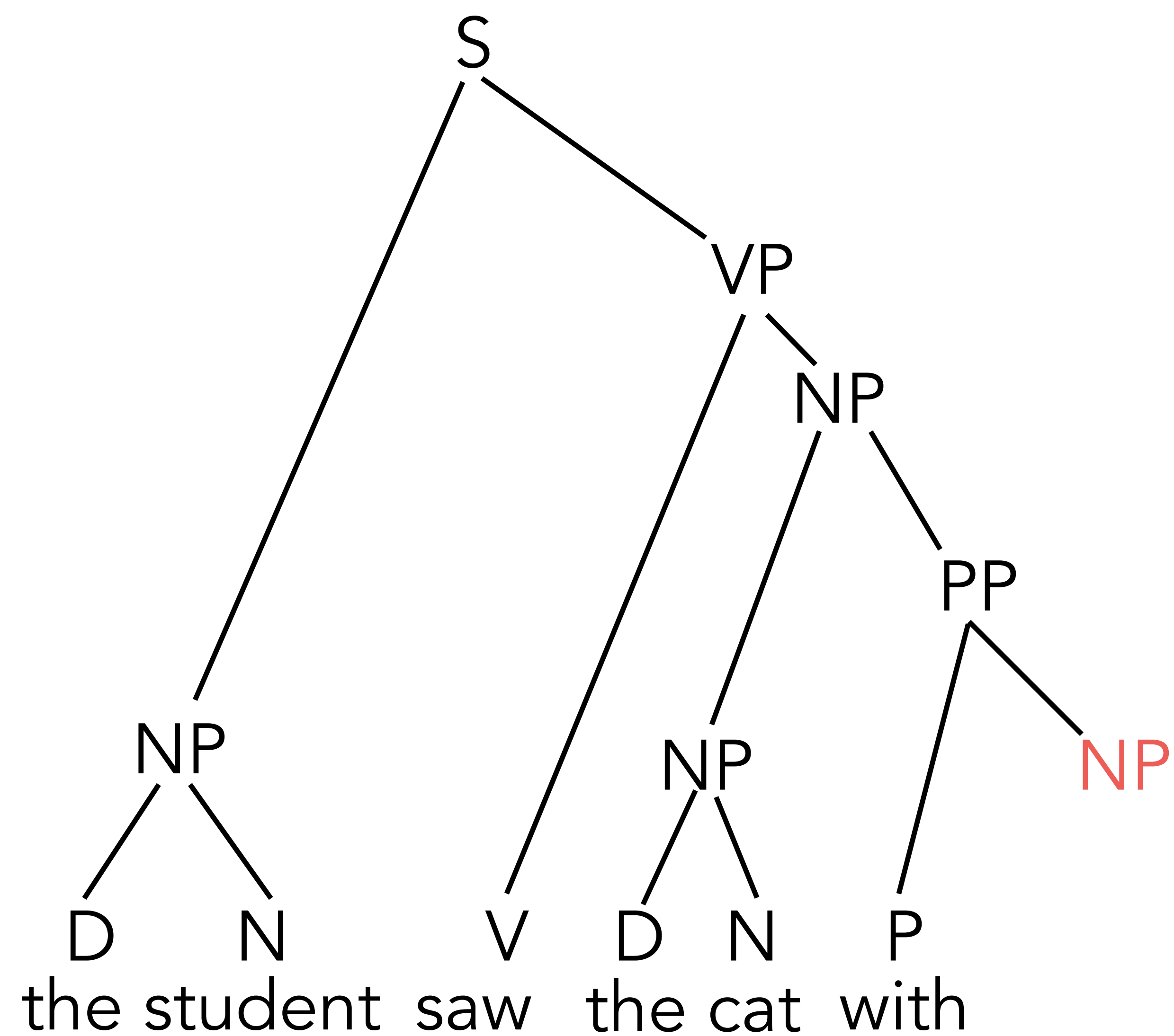
S → NP VP	V → saw
VP → V NP	P → with
VP → VP PP	D → the
PP → P NP	N → cat
NP → D N	N → tail
NP → NP PP	N → student



Context Free Grammars (CFG)

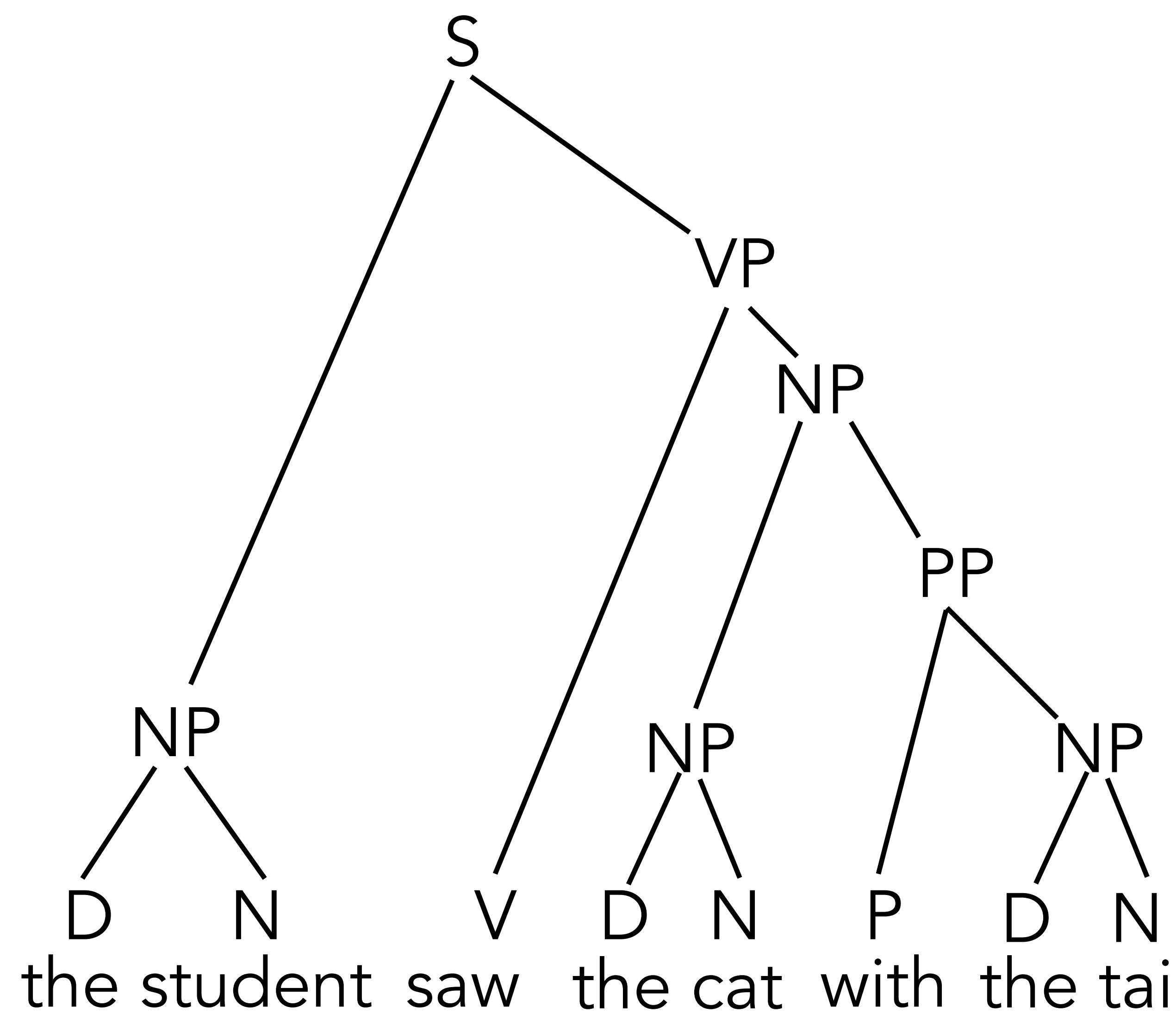
S → NP VP
VP → V NP
VP → VP PP
PP → P NP
NP → D N
NP → NP PP

V → saw
P → with
D → the
N → cat
N → tail
N → student



Context Free Grammars (CFG)

S	→	NP VP	V	→	saw
VP	→	V NP	P	→	with
VP	→	VP PP	D	→	the
PP	→	P NP	N	→	cat
NP	→	D N	N	→	tail
NP	→	NP PP	N	→	student



Context Free Grammars

- A context free grammar is defined by:
  - Set of **terminal symbols** Σ.
  - Set of **non-terminal symbols** N.
  - A **start symbol** $S \in N$.
  - Set R of **productions** of the form $A \rightarrow \beta$,
    where A $\in$ N and $\beta \in (\Sigma \cup N)^*$, *i.e.* β is a string of terminals and non-terminals.

Language of a CFG

- Given a CFG $G=(N, \Sigma, R,S)$:
  - Given a string $\alpha A\gamma$, where $A \in N$, we can derive $\alpha\beta\gamma$ if there is a production $A \rightarrow \beta \in R$.
  - $\alpha \Rightarrow \beta$ means that G can derive β from α in a single step.
  - $\alpha \Rightarrow^* \beta$ means that G can derive β from α in a finite number of steps.
  - The **language of G** is defined as the set of all terminal strings that can be derived from the start symbol.

$$L(G) = \{\beta \in \Sigma^*, \text{ s.t. } S \Rightarrow^* \beta\}$$

Derivations and Derived Strings

- CFG is a string rewriting formalism, so the **derived objects** are strings.

- A derivation is a sequence of rewriting steps.

- CFGs are **context free:** applicability of a rule depends only on the nonterminal symbol, not on its context.

  - Therefore, the order in which multiple non-terminals in a partially derived string are replaced does not matter.
    We can represent identical derivations in a **derivation tree.**

  - The derivation tree *implies* a parse tree.

Recursion in CFGs

S	→ NP VP		V	→ saw
VP	→ V NP		P	→ with
VP	→ VP PP		D	→ the
PP	→ P NP		N	→ cat
NP	→ D N		N	→ tail
NP	→ NP PP		N	→ student

Parse Tree:

NP

Derived String:

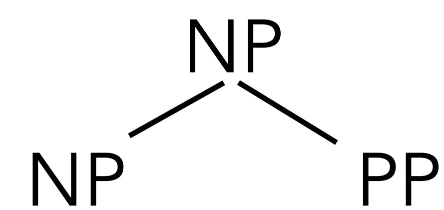
NP

Recursion in CFGs

S → NP VP
VP → V NP
VP → VP PP
PP → P NP
NP → D N
NP → NP PP

V → saw
P → with
D → the
N → cat
N → tail
N → student

Parse Tree:



Derived String:

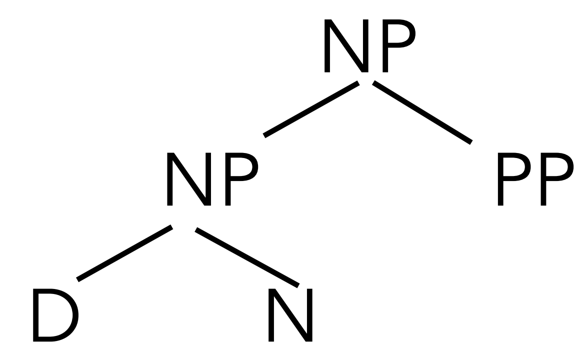
NP PP

Recursion in CFGs

S → NP VP
VP → V NP
VP → VP PP
PP → P NP
NP → D N
NP → NP PP

V → saw
P → with
D → the
N → cat
N → tail
N → student

Parse Tree:



Derived String:

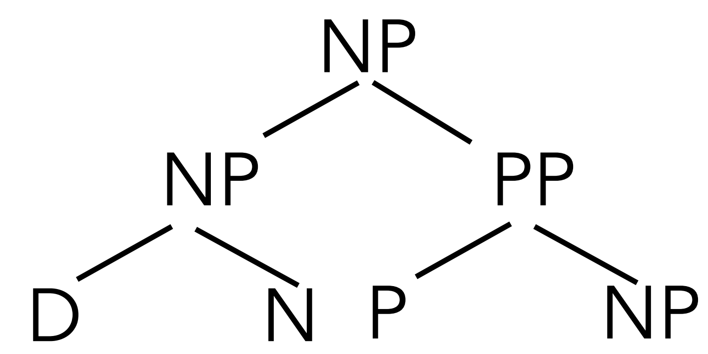
the student PP

Recursion in CFGs

S → NP VP
VP → V NP
VP → VP PP
PP → P NP
NP → D N
NP → NP PP

V → saw
P → with
D → the
N → cat
N → tail
N → student

Parse Tree:



Derived String:

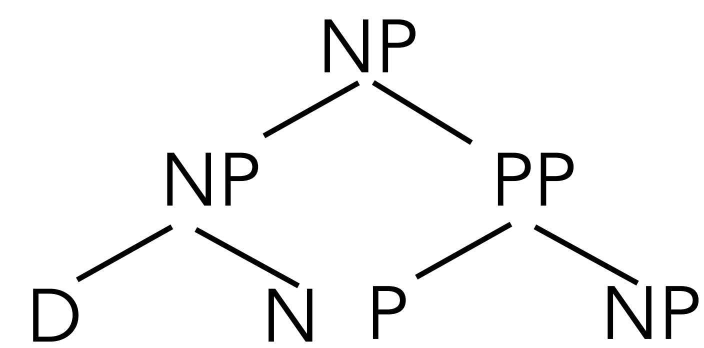
the student P NP

Recursion in CFGs

S → NP VP
VP → V NP
VP → VP PP
PP → P NP
NP → D N
NP → NP PP

V → saw
P → with
D → the
N → cat
N → tail
N → student

Parse Tree:



Derived String:

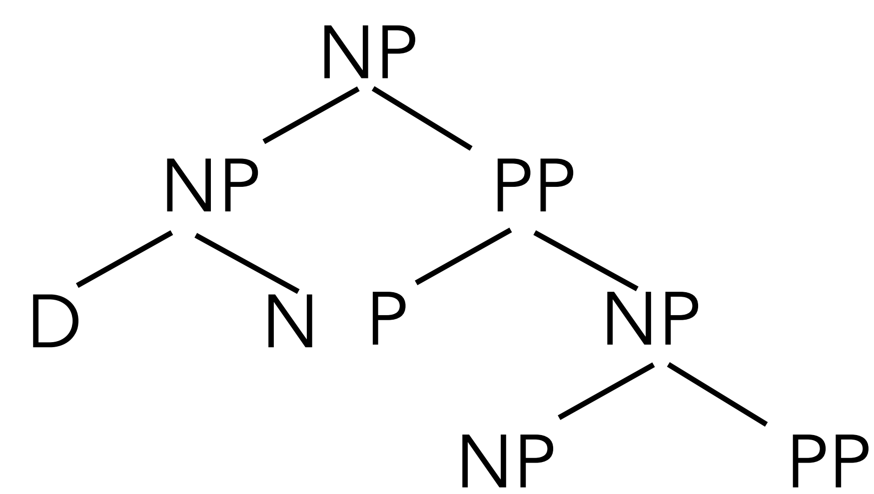
the student with NP

Recursion in CFGs

S → NP VP
VP → V NP
VP → VP PP
PP → P NP
NP → D N
NP → NP PP

V → saw
P → with
D → the
N → cat
N → tail
N → student

Parse Tree:



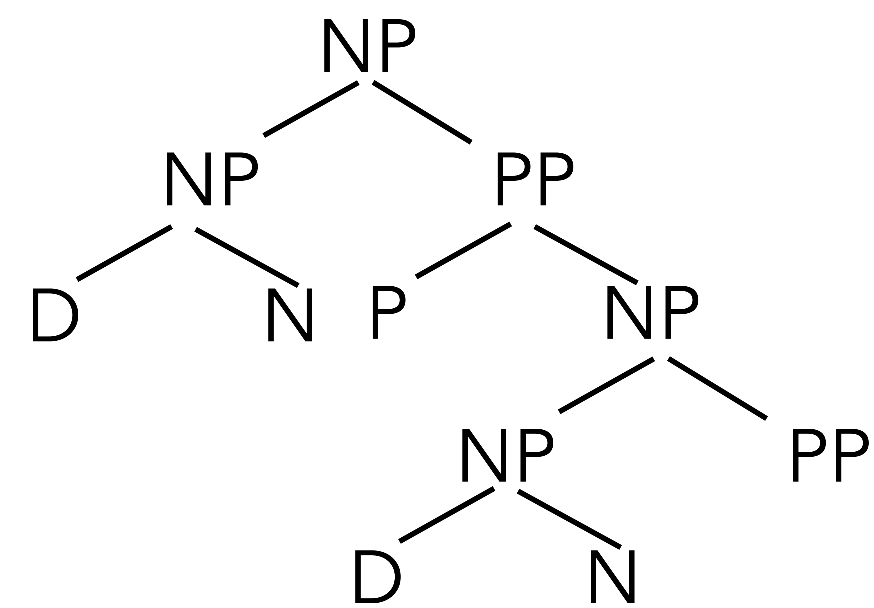
Derived String:

the student with NP PP

Recursion in CFGs

S → NP VP		V → saw	
VP → V NP		P → with	
VP → VP PP		D → the	
PP → P NP		N → cat	
NP → D N		N → tail	
NP → NP PP		N → student	

Parse Tree:



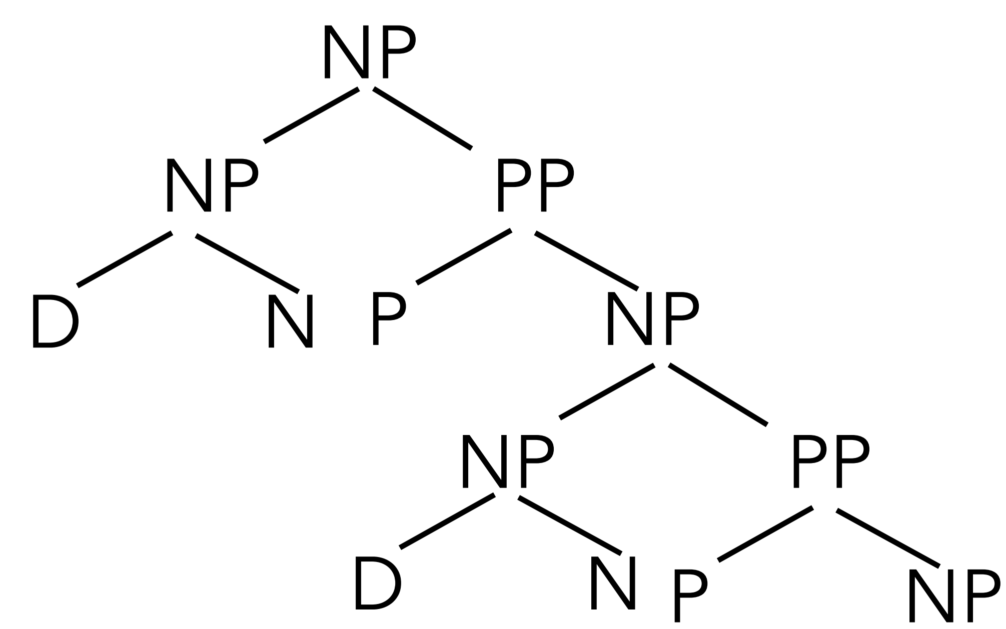
Derived String:

the student with the cat PP

Recursion in CFGs

S	→ NP VP	V	→ saw
VP	→ V NP	P	→ with
VP	→ VP PP	D	→ the
PP	→ P NP	N	→ cat
NP	→ D N	N	→ tail
NP	→ NP PP	N	→ student

Parse Tree:



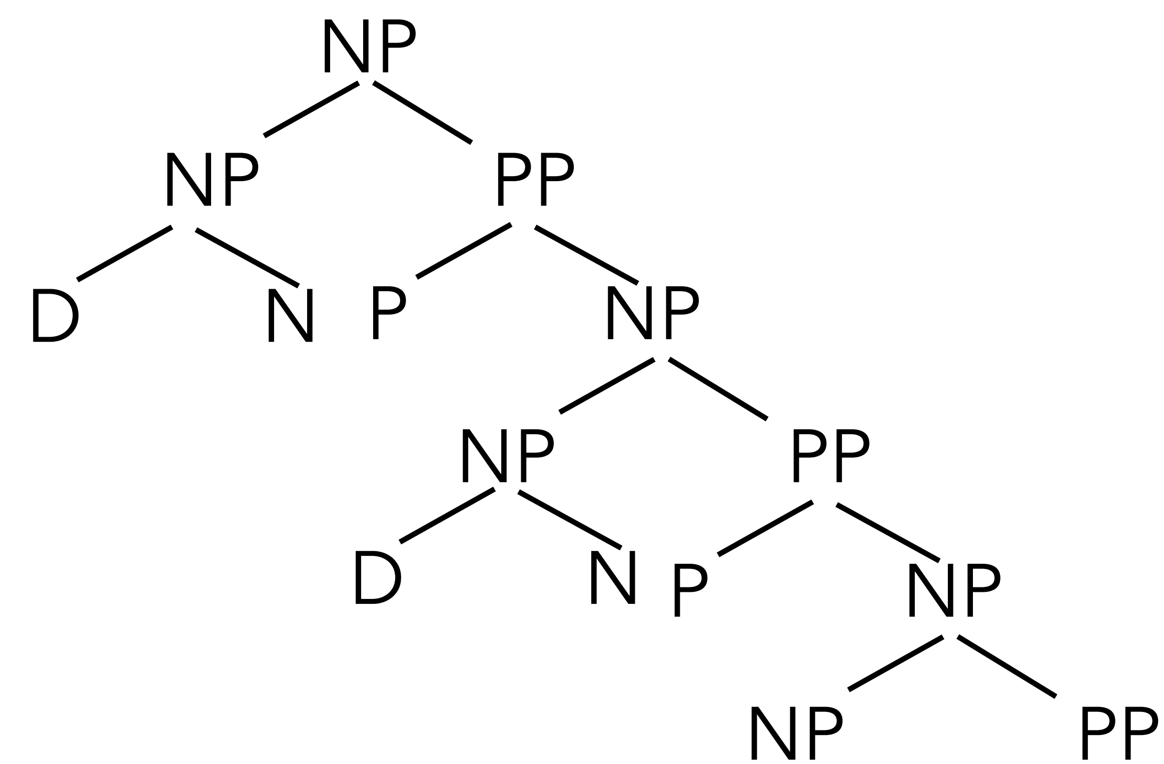
Derived String:

the student with the cat with NP

Recursion in CFGs

S → NP VP	V → saw
VP → V NP	P → with
VP → VP PP	D → the
PP → P NP	N → cat
NP → D N	N → tail
NP → NP PP	N → student

Parse Tree:



Derived String:

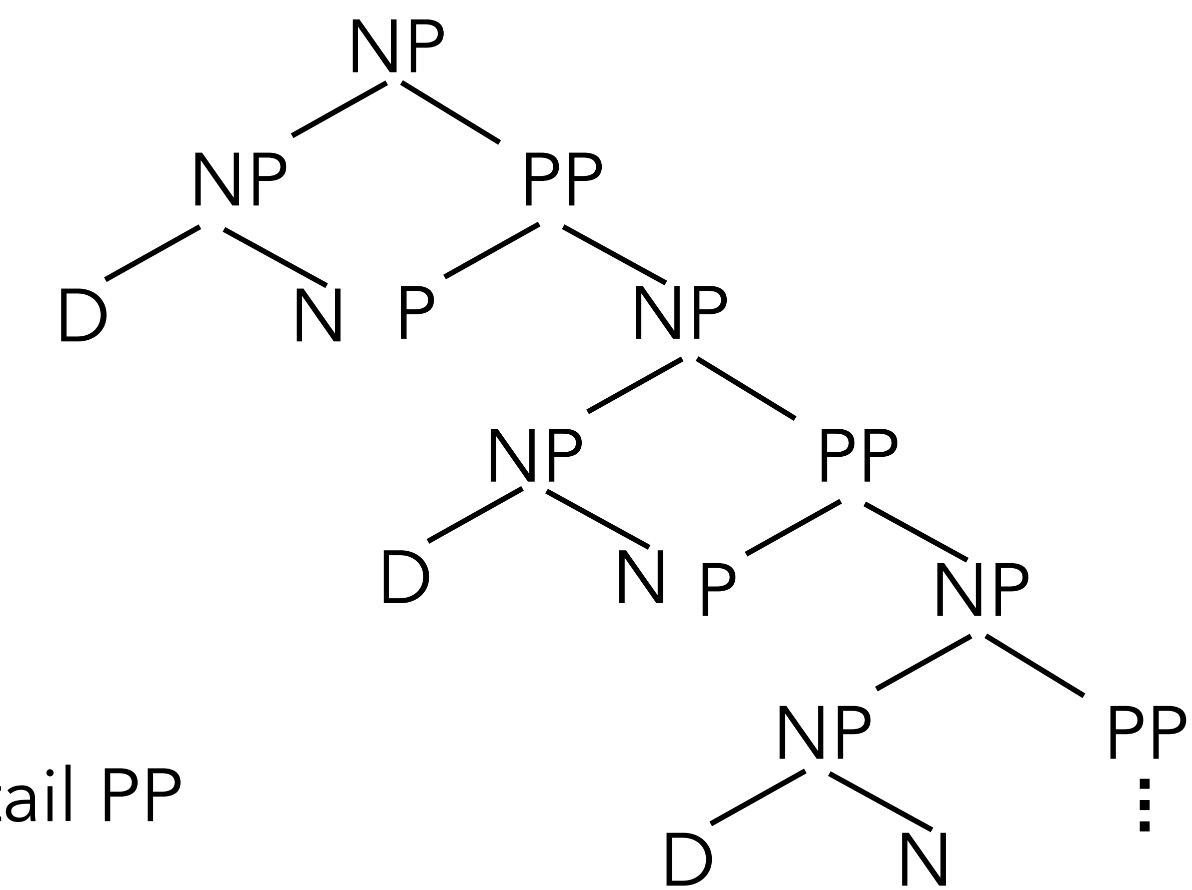
the student with the cat with NP PP

Recursion in CFGs

S → NP VP
VP → V NP
VP → VP PP
PP → P NP
NP → D N
NP → NP PP

V → saw
P → with
D → the
N → cat
N → tail
N → student

Parse Tree:



Derived String:

the student with the cat with the tail PP

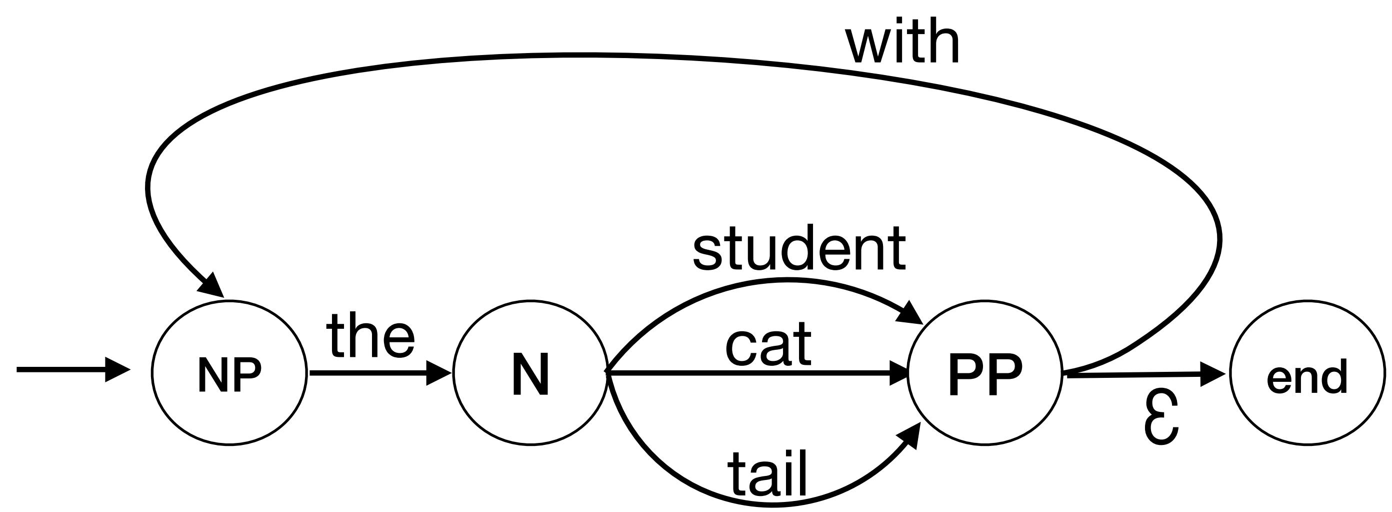
Regular Grammars

- A regular grammar is defined by:
  - Set of **terminal symbols** Σ.
  - Set of **non-terminal symbols** N.
  - A **start symbol** $S \in N$.
  - Set R of **productions** of the form $A \to aB$, or $A \to a$ where $A, B \in N$ and $a \in \Sigma$.

Finite State Automata

- Regular grammars can be implemented as finite state automata.

NP → the N
N → student PP
N → cat PP
N → tail PP
PP → with NP
PP → ε

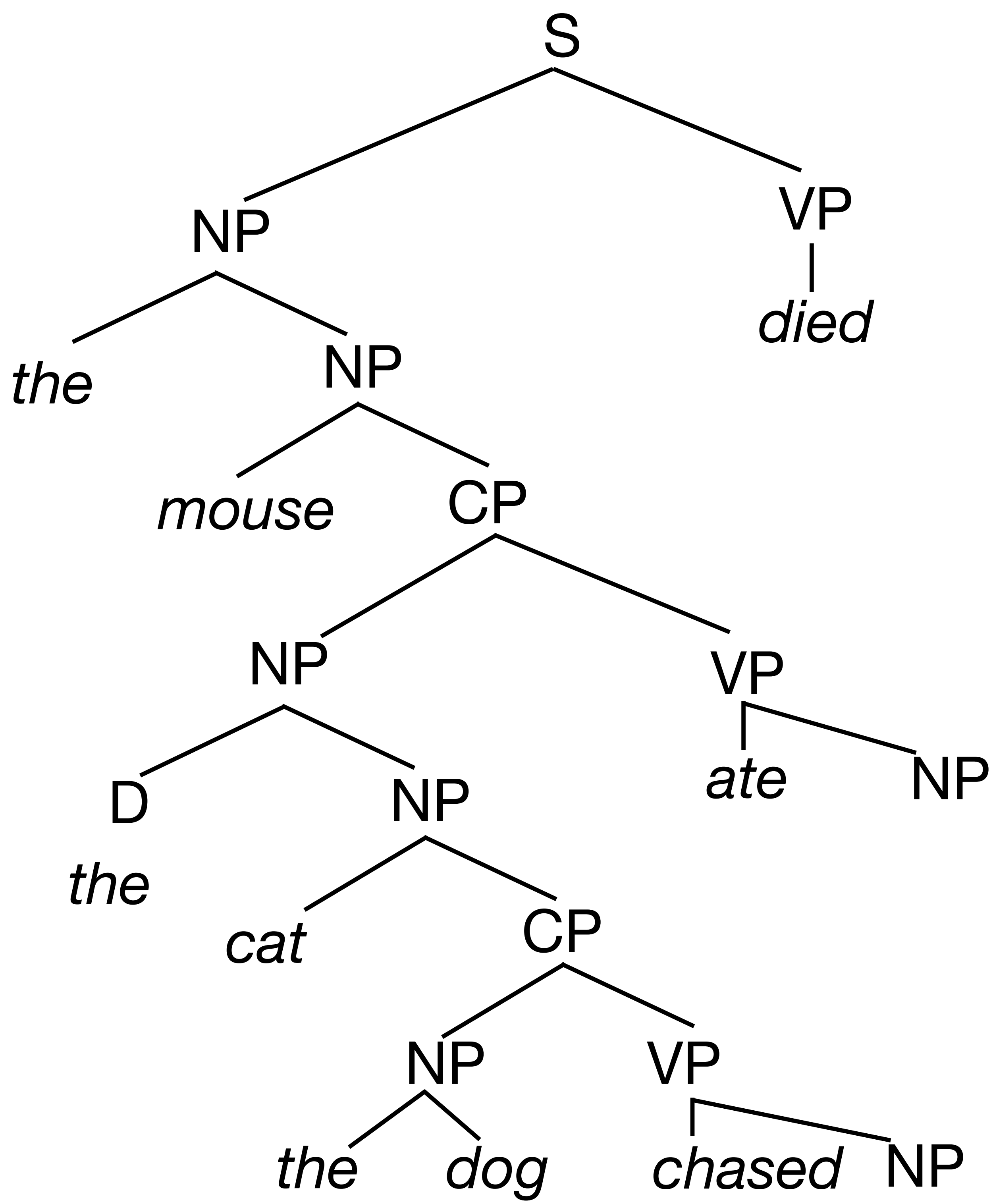


- The set of all regular languages is strictly smaller than the set of context-free languages.

Are natural languages (such as English, specifically) regular?

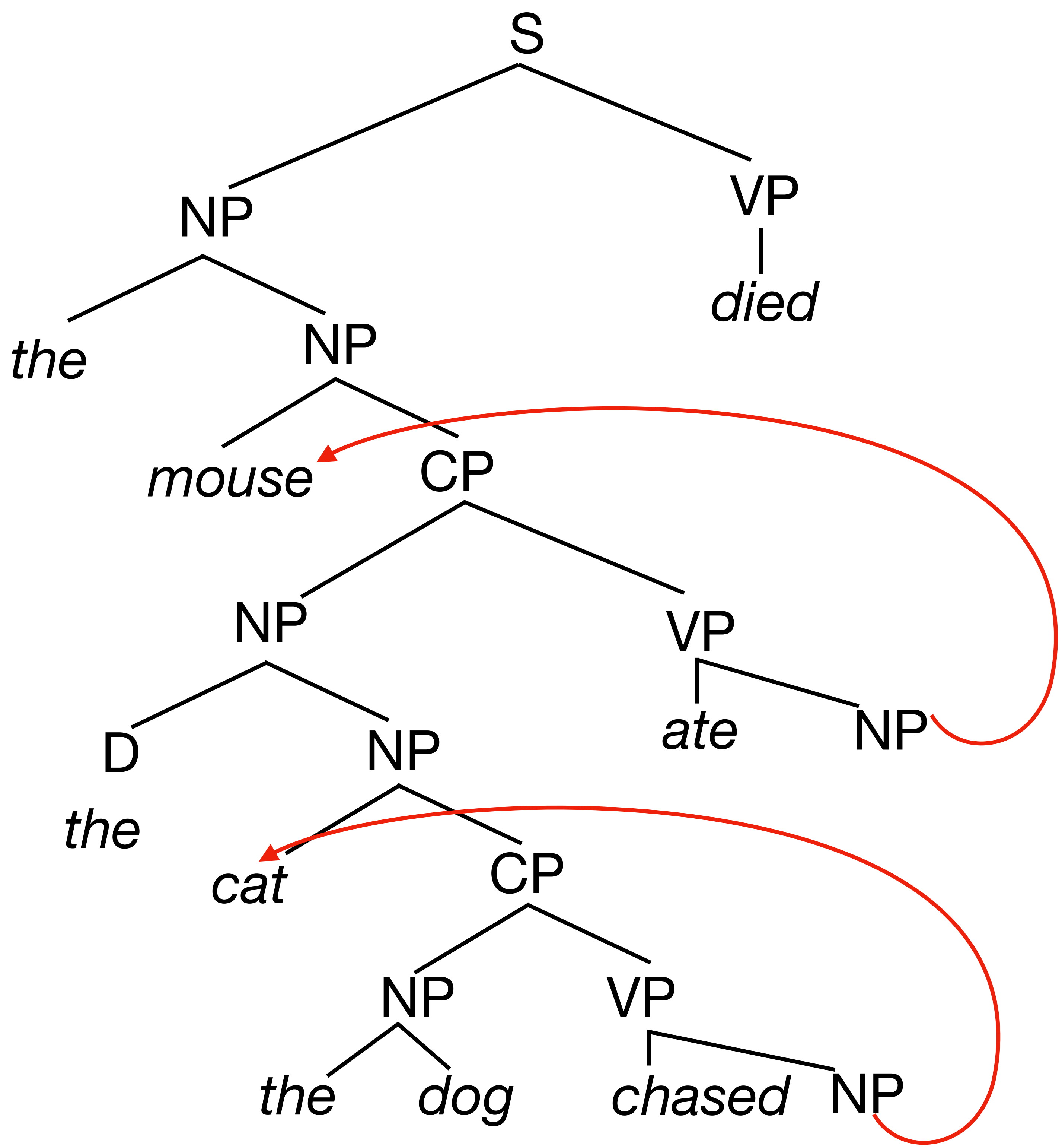
No! Example: Center embeddings.

Center Embeddings



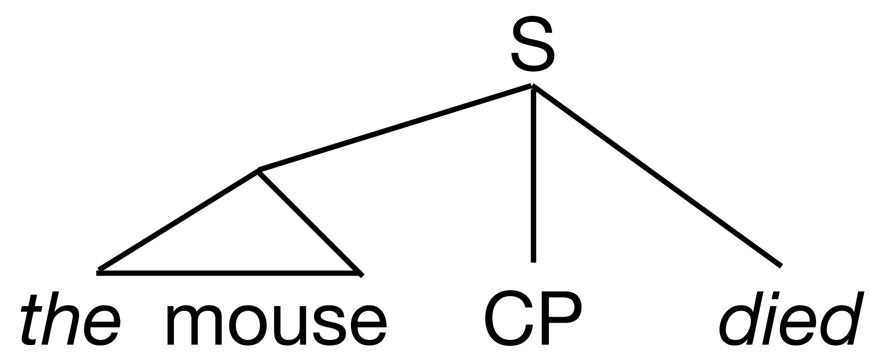
Linguistically, this is not a perfect analysis.

Center Embeddings



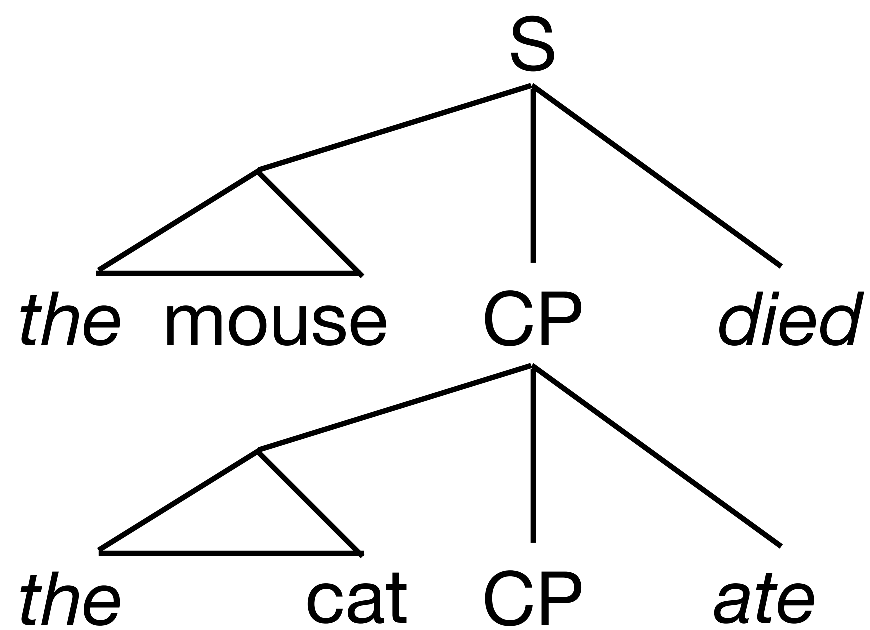
Linguistically, this is not a perfect analysis.

Center Embeddings



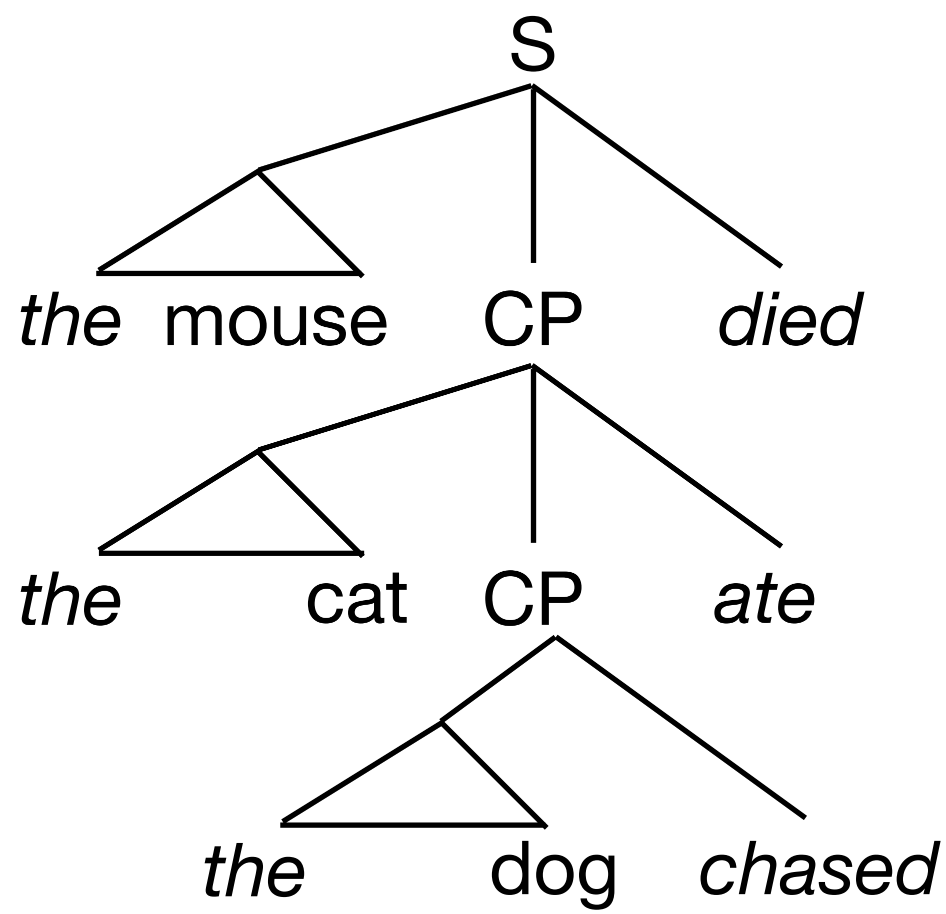
Linguistically, this is not a perfect analysis.

Center Embeddings



Linguistically, this is not a perfect analysis.

Center Embeddings



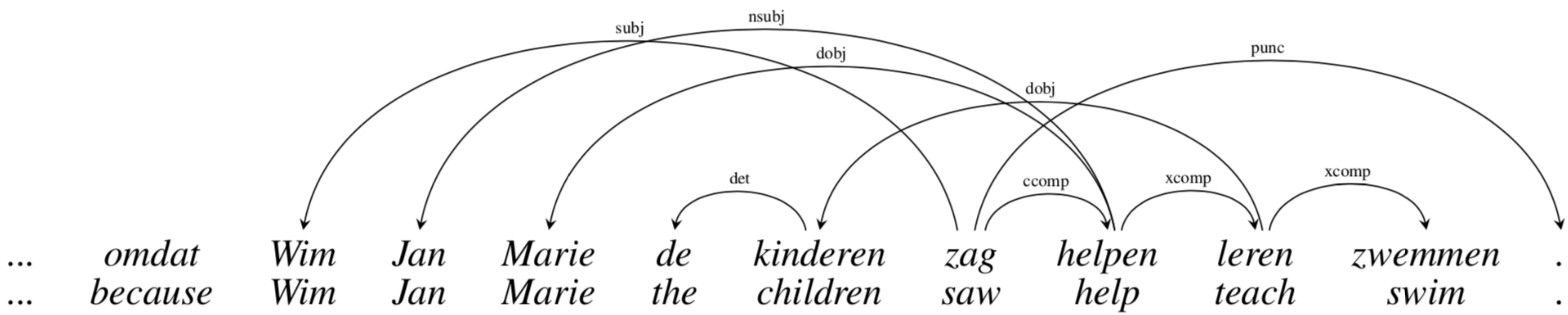
Problem: Regular grammars cannot capture long-distance dependencies.

This example follows the pattern $\mathbf{a^n b^n}$.
Can show that is language is not regular (using the "pumping lemma").

Linguistically, this is not a perfect analysis.

Is Natural Language Context Free?

Probably not. An example from Dutch:



"…because Wim saw Jan help Marie teach the children to swim"

Context Free Grammars cannot describe crossing dependencies.
For example, it can be shown that

$$\mathbf{a^n b^m c^n d^m}$$

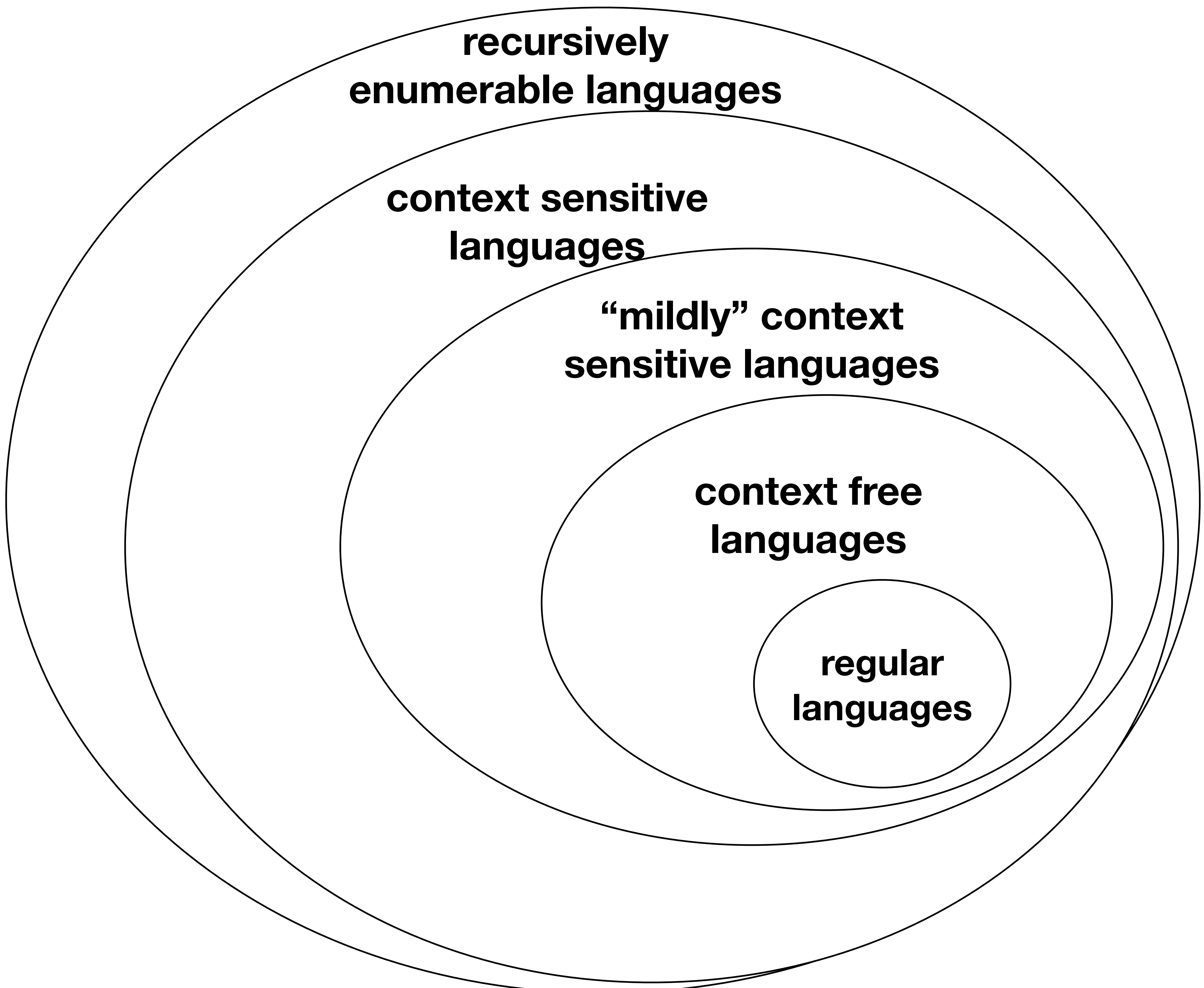
is not a context free language.

Complexity Classes

- Regular languages. (regular grammars, regular expressions, finite state automata)
- Context-free languages. (context-free string grammars, pushdown automata).
- “Mildly-context-sensitive languages”.
- Context-sensitive languages.
- Recursively enumerable languages (Turing machines).

This is sometimes called the “Chomsky Hierarchy”.

Complexity Classes



Formal Grammar and Parsing

- Formal Grammars are used in linguistics, NLP, programming languages, …
- We want to build a compact model that describes a complete language.
- Need efficient algorithms to determine if a sentence is in the language or not **(recognition problem)**.
- We also want to recover the structure imposed by the grammar (parsing problem).