Introduction to *syntax* and *parsing*

Lecture 3

CSC485/2501

Syntactic structure

- *Syntax*:
  - The combinatorial structure of words.
  - How words can be linearly organized:
    - ***Left/right precedence*.**
    - ***Contiguity***.
  - How words can be hierarchically organized into ***phrases*** and ***sentences***.

Syntactic structure

- *Syntax*:

The cat hunted the squirrel living in the tree with persistence.

vs.

Squirrel persistence in cat the living tree with the hunted the.

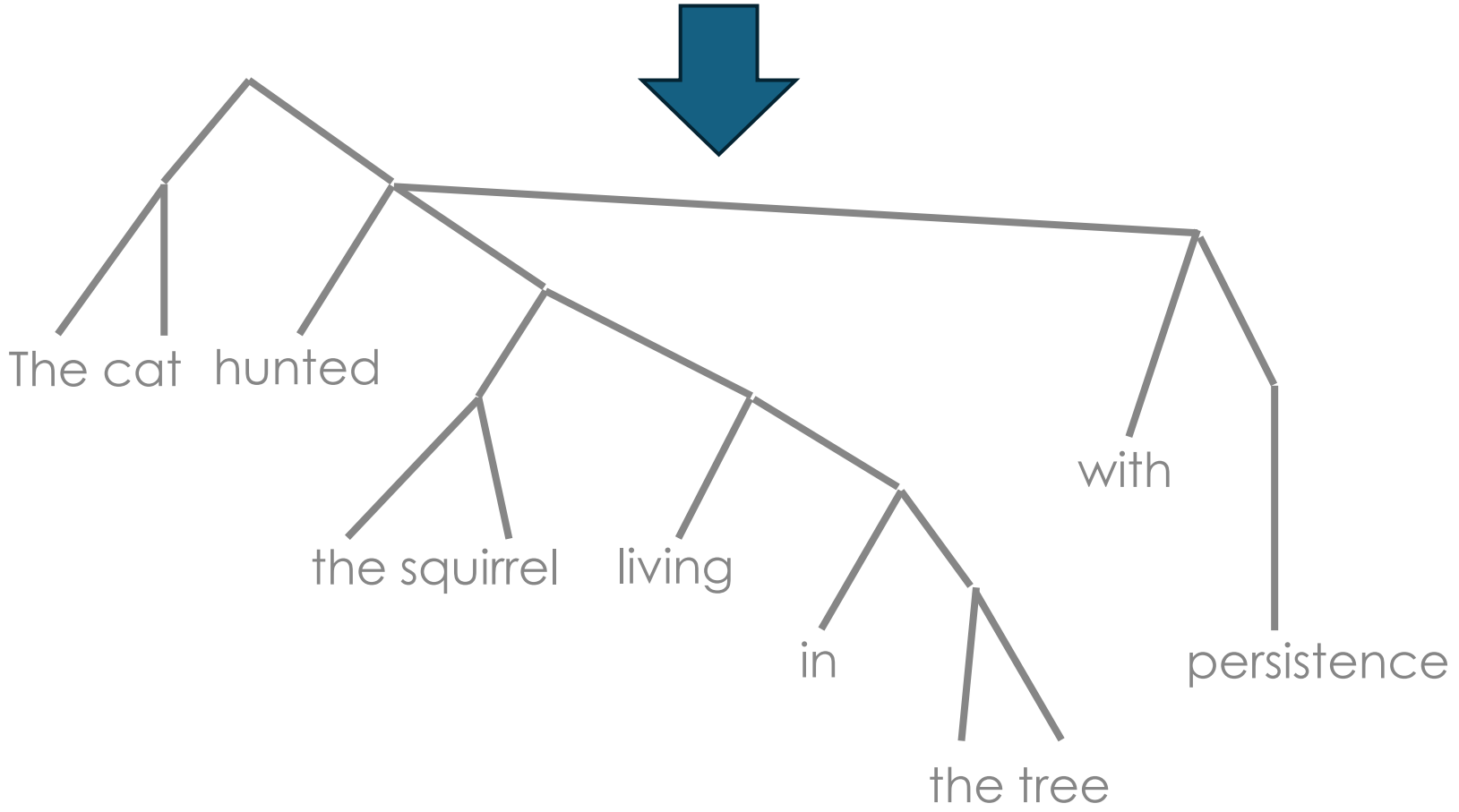
Syntactic structure

The cat hunted the squirrel living in the tree with persistence.

```
[[The cat]
 [hunted [the squirrel [living [in [the tree]]]]
 [with [persistence]]]]
```

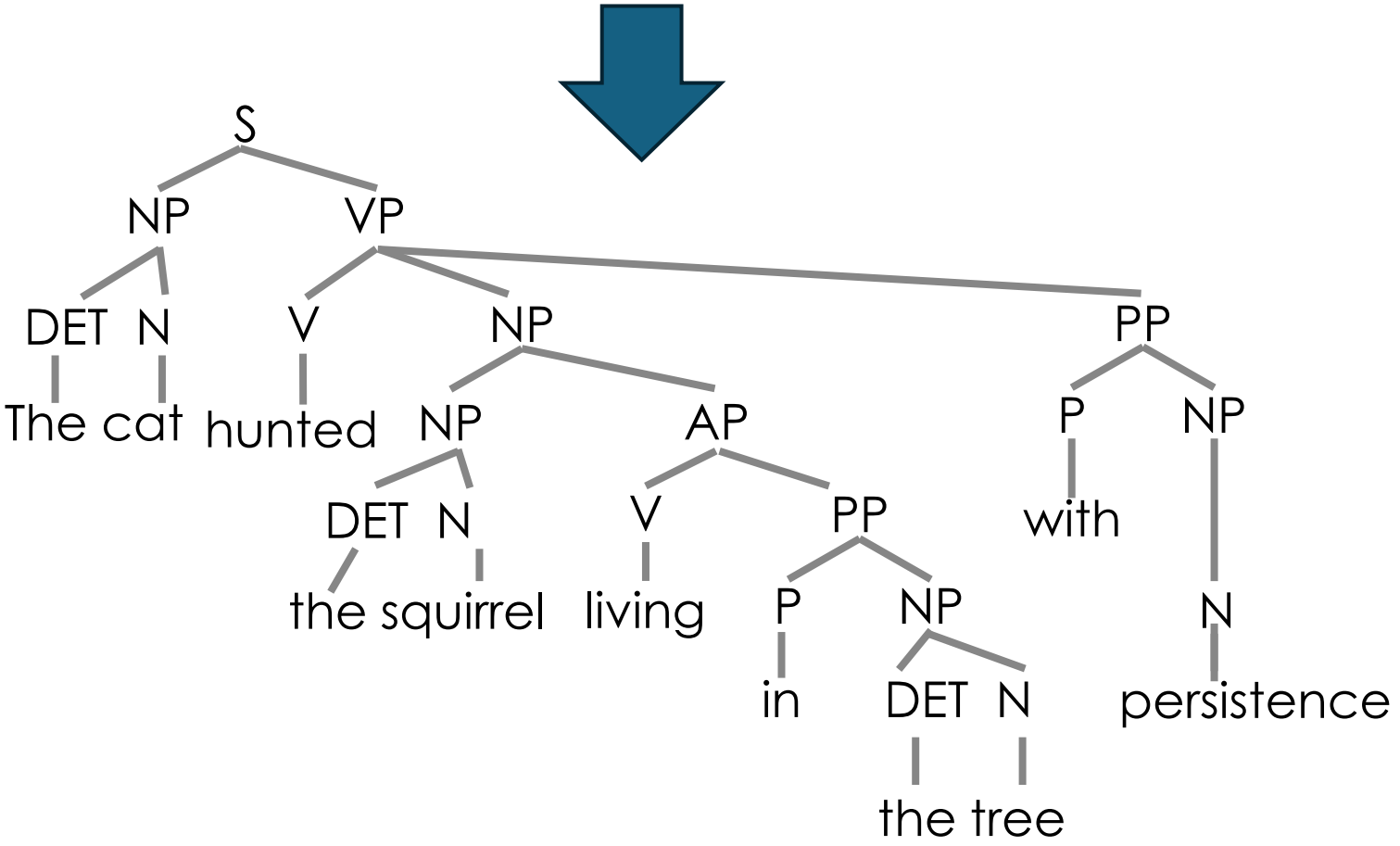
Syntactic structure

The cat hunted the squirrel living in the tree with persistence.



Syntactic structure

The cat hunted the squirrel living in the tree with persistence.



Grammars and Parsing

- Grammar:
  - Formal specification of allowable structures.
    - Knowledge
    - Representation
- Parsing:
  - Analysis of string of words to determine the structure assigned by grammar.
    - Algorithm
    - Process

Using grammar to capture structure

- Main issues:
  - Which words are grouped together into phrases.
  - How words within a phrase project the properties of a single, common **word** (the **head** of the phrase).
  - How different phrases relate to each other.

- Grammar encodes these relations. Some grammars interpret these relations with respect to meaning.

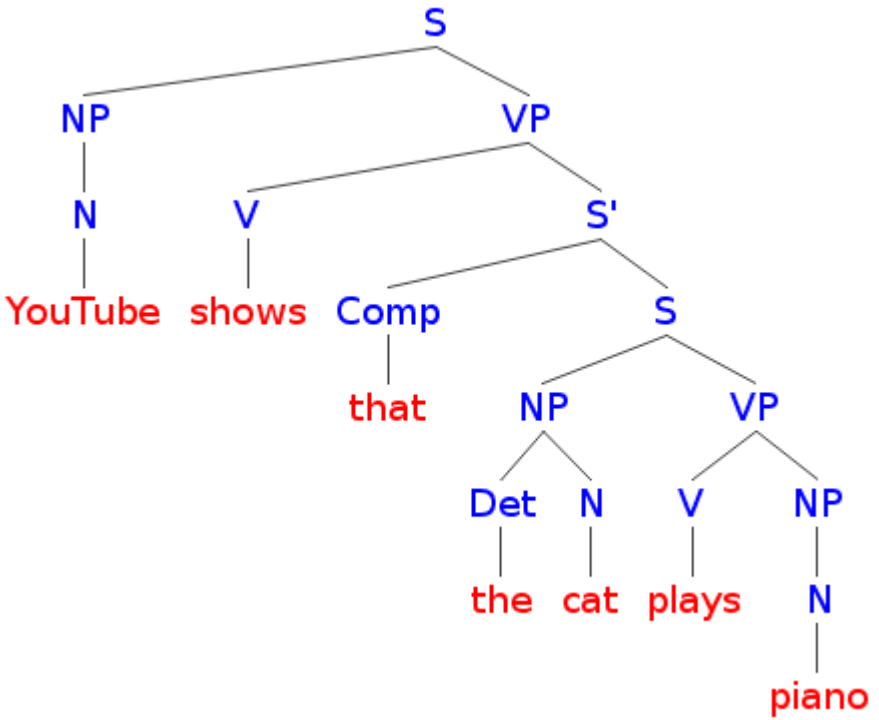
Good and Bad Grammars

- There are many possible grammars for any natural language.
  - Some are better than others.
- Desiderata:
  - Faithfulness to (vastly divergent) details about language.
  - Economy of description.
  - Fidelity to some prevailing linguistic intuition.
  - Efficiency of parsing.

Elements of Grammar

Primitives

Combinations



Elements of Grammar: *Primitives*

- Primitives: lexical categories or parts of speech.
  - Each **word-type** is a member of one or more.
  - Each **word-token** is an instance of exactly one.

  e.g. The cat in the hat sat. -> 6 tokens, 5 types.
- Categories are **open** or **closed** to new words.
  - *Function word* and *content word*.
- Eight main categories, many subcategories.

  ~~Nine~~ ~~Seven~~

  Twenty-three?
- The categories might possibly be language-specific as well.

Type–token distinction

Parts of Speech

- **Nouns**: denote an object, a concept, a place, …
  - **Count nouns**: *dog, shoe, Band-Aid, …*
  - **Mass nouns**: *water, wheat, …*
  - **Proper nouns**: *Fred, New York City, …*
- Pronouns: *he, she, you, I, they, …*
- Adjectives: denote an attribute of the denotation of a noun.
  - Intersective: *pink, furry, …*
  - Measure: *big, small, …*
  - Intensional: *former, alleged, …*

Parts of Speech

- Verbs: predicates, denote an action or a state.
  Numerous distinctions, e.g. transitivity:
  - Intransitive: *sleep, die, …*
  - Transitive: *eat, kiss, …*
  - Ditransitive: *give, sell, …*
  - Copula: *be, feel, become, …*
- Determiners, articles: specify certain attributes of the denotation of a noun that are grammatically relevant
  - *the, a, some, …*

Parts of Speech

- Adverbs: denote an attribute of the denotation of a predicate.
  - Time and place: *today, there, now, …*
  - Manner: *happily, furtively, …*
  - Degree*: *much, very, …*
- Prepositions: relate two phrases with a location, direction, manner, etc.
  - *up, at, with, in front of, before, …*

Parts of Speech

- Conjunctions: combine two clauses or phrases:
  - Coordinating conjunctions: *and, or, but,* …
  - Subordinating conjunctions: *because, while,* …
- Interjections: stand-alone emotive expressions:
  - *um, wow, oh dear,* …

Elements of Grammar: *Combinations*

- ***Combinations***:
  - **Phrase**: a hierarchical grouping of words and/or phrases.
  - **Clause**: a phrase consisting of a verb and (almost) all its dependents.
  - **Sentence**: a clause that is syntactically independent of other clauses.
- Can be represented by tree (or a labelled bracketing).
- Terminology: A ***constituent*** is a well-formed phrase with overtones of semantic and/or psychological significance.

Types of Phrase

- Noun phrase (NP):
  - *a mouse*
  - *mice*
  - *Mickey*
  - *the handsome marmot*
  - *the handsome marmot on the roof*
  - *the handsome marmot whom I adore*

- Verb phrase (VP):
  - *laughed loudly*
  - *give the book to Mary*
  - *slept*
  - *quickly gave the book to Mary*

Types of Phrase

- Adjective phrase (AP, AdjP):
  - *green*
  - *proud of Kyle*
  - *very happy that you went*

- Prepositional phrase (PP):
  - *in the sink*
  - *without feathers*
  - *astride the donkey*

Clauses and Sentences

- Clauses:
  - *Ross remarked upon Nadia's dexterity*
  - *to become a millionaire by the age of 30*
  - *that her mother had lent her for the banquet*

- Sentences:
  - *Ross remarked upon Nadia's dexterity.*
  - *Nathan wants to become a millionaire by the age of 30.*
  - *Nadia rode the donkey that her mother had lent her for the banquet.*
  - *The handsome marmot on the roof [in dialogue].*

Clauses and Sentences

- Clauses may act as phrases.
- NP:
  - To become a millionaire by the age of 30 is what Ross wants.
  - Nadia riding her donkey is a spectacular sight.
  - Ross discovered that Nadia had been feeding his truffles to the donkey.

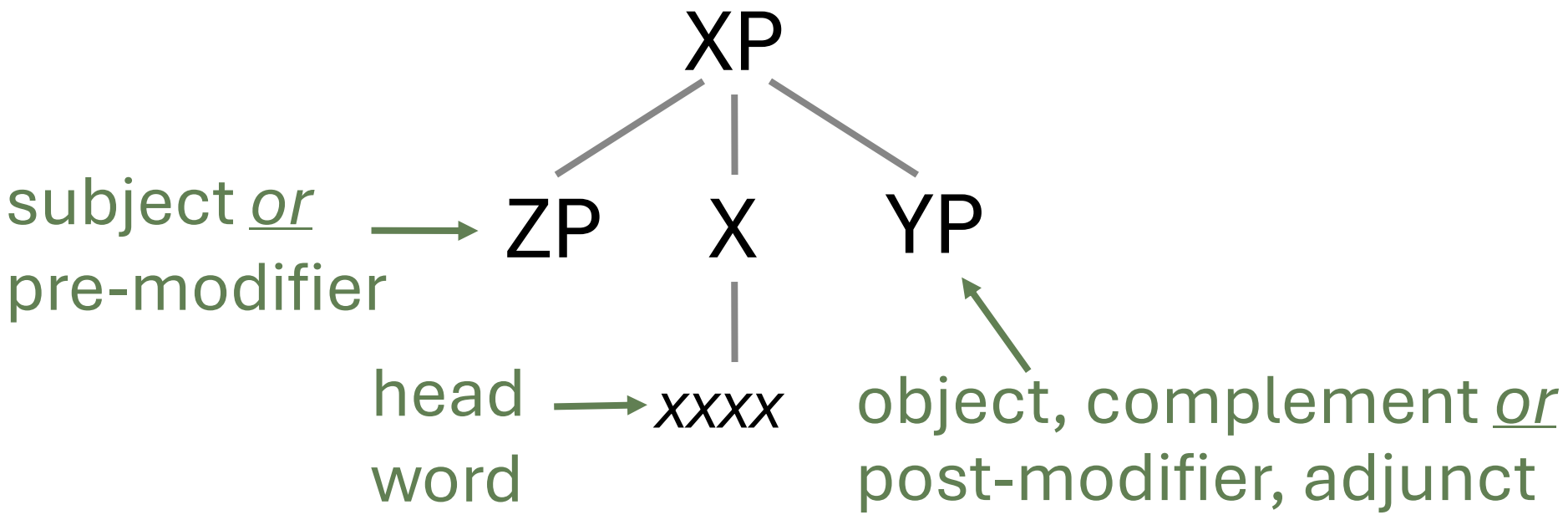
Quiz

- The word that shares properties such as number with its phrase (and from which the phrase takes its name) is called the phrase's:

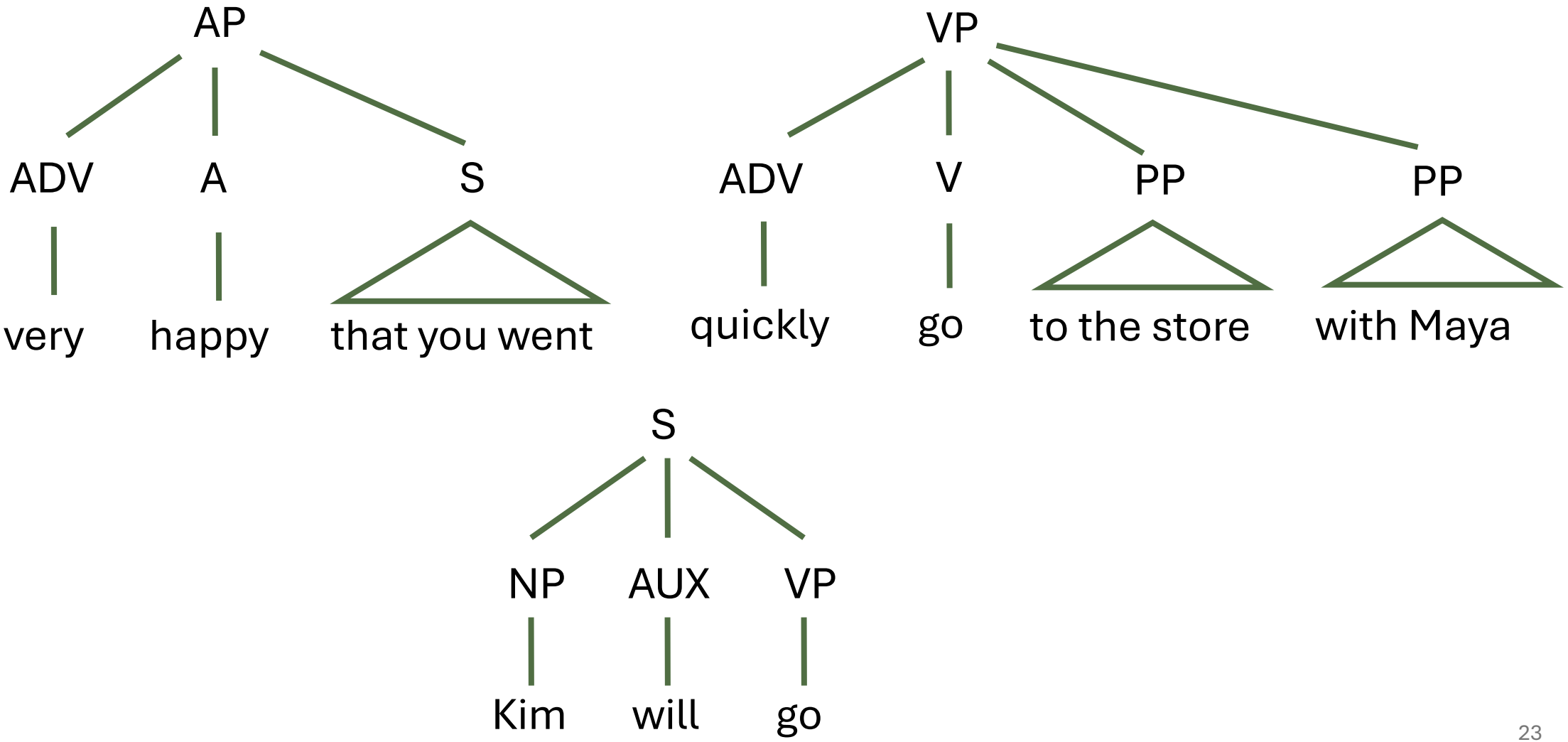
A. foot
B. head
C. centre
D. morpheme

The structure of an idealized phrase

XP → ZP X YP



Example phrases



Formal Definition of a CFG

- A context-free grammar is a quadruple $G = (V_t, V_n, P, S)$, where
- V_t is a finite set of terminal symbols. (tokens)
- V_n is a finite set of non-terminal symbols. (phrase, clause, S)
- P is a finite set of production rules of the form
  $A \rightarrow \alpha$
  where $A \in V_n$ and α is a sequence of symbols in $(V_n \cup V_t)^*$.
- $S \in V_n$ is the start symbol.

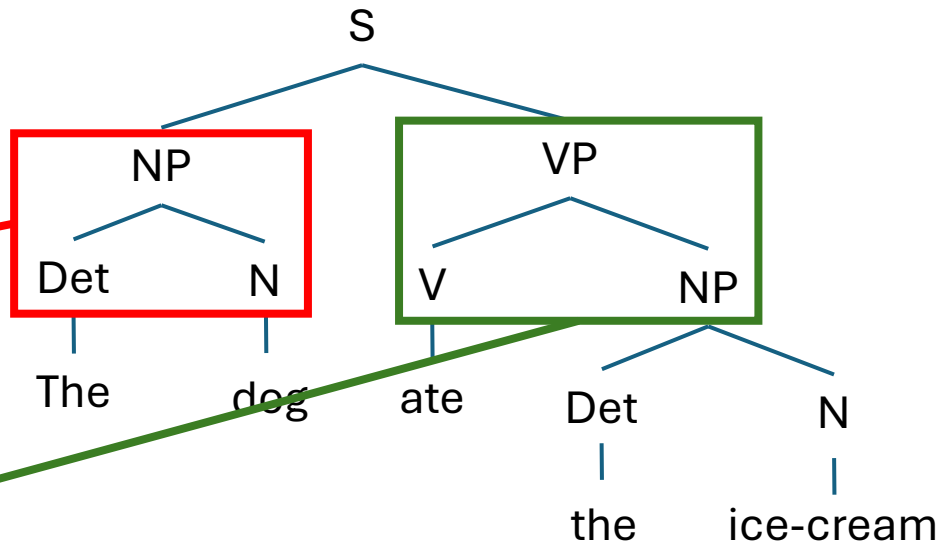
$S=S$, $P=\{$

```
S   → NP   VP
NP  → Det   N
NP  → Det   Adj   N
NP  → NP   PP
VP  → V
VP  → V   NP
PP  → P   NP
Det → the | a | an
Adj → old | red | happy | …
N   → dog | park | ice-cream | contumely | run | …
V   → saw | ate | run | disdained | …
P   → in | to | on | under | with | … }
```

V_t and V_n can be inferred from the production rules.

The lexicon:
In practice, a separate data structure

Lexical categories:
NT's that rewrite as a single T.



$S=S,\ P=\{$

S → NP VP

NP → Det N

NP → Det Adj N

NP → NP PP

VP → V

VP → V NP

PP → P NP

Det → *the* | *a* | *an*

Adj → *old* | *red* | *happy* | …

N → *dog* | *park* | *ice-cream* | *contumely* | *run* | …

V → *saw* | *ate* | *run* | *disdained* | …

P → *in* | *to* | *on* | *under* | *with* | … }

Terminology

- **Non-terminal** (NT):
  A symbol that occurs on the left-hand side (lhs) of some rule.
- **Pre-terminal**:
  a kind of non-terminal located on the LHS of a lexical entry.
- **Terminal** (T):
  A symbol that never occurs on the lhs of a rule.
- **Start symbol**:
  A specially designated NT that must be the root of any tree derived from the grammar.
  In our grammars, it is usually S for sentence.

Parsing

- Parsing: Determining the structure of a sequence of words, given a grammar.
  - Which grammar rules should be used?
  - To which symbols (words / terminals and nodes / non-terminals) should each rule apply?

Parsing

- Input:
  - A context-free grammar.
  - A sequence of words

    Time flies like an arrow.

  or, more precisely, of sets of parts of speech.

    {*noun,verb*} {*noun,verb*} {*verb,prep*} {*det*} {*noun*}
- Process:
  - (Working from left to right?,) ***guess*** how each word fits in.

Top-down Parsing

- ***Top-down*** or ***rule-directed*** parsing:
  "Can I take these rules and match them to this input?"
  - Initial goal is an S.
  - Repeatedly look for rules that decompose /expand current goals and give new goals.
    E.g., goal of S may decompose to goals NP and VP.
  - Eventually get to goals that look at input.
    E.g., goal of NP may decompose to det noun.
  - Succeed iff entire input stream is accounted for as S.

Top-down Parsing

- Example: A recursive descent parser.
  `>>> nltk.app.rdparser()`
- Operations on leftmost frontier node:
  - Expand it.
  - Match it to the next input word.
- Choice of next operation (in nltk demo):
  - If it's a terminal, try matching it to input.
  - If it's a non-terminal, try expanding with first-listed untried rule for that non-terminal.

Bottom-up Parsing

- ***Bottom-up*** or ***data-directed*** parsing:
  "Can I take this input and match it to these rules?"
  - Try to find rules that match a possible PoS of the input words …
  - … and then rules that match the constituents so formed.
  - Succeed iff the entire input is eventually matched to an S.

Bottom-up Parsing

- Example: A shift–reduce parser.
  `>>> nltk.app.srparser()`
- Operations:
  - ***Shift*** next input word onto stack.
  - Match the top n elements of stack to **RHS** of rule, ***reduce*** them to **LHS**.

- Choice of next operation (in `nltk` demo):
  - Always prefer reduction to shifting.
  - Choose the first-listed reduction that applies.
- Choice of next operation (in real life):
  - Always prefer reduction to shifting for words, but not necessarily for larger constituents.

Problems

- Neither top-down nor bottom-up search exploits useful idiosyncrasies that CFG rules, alone or together, often have.
- Problems:
  - Recomputation of constituents.
  - Recomputation of common prefixes.
- Solution: Keep track of:
  - Completed constituents.
  - Partial matches of rules.

Quiz

The word that shares properties such as number with its phrase (and from which the phrase takes its name) is called the phrase's:

A. foot
B. head
C. centre
D. morpheme