ANTONIS ANASTASOPOULOS
CS499 INTRODUCTION TO NLP

SYNTAX

https://cs.gmu.edu/~antonis/course/cs499-spring21/

With adapted slides by David Mortensen and Alan Black

STRUCTURE OF THIS LECTURE

1. What is Syntax
2. Constituency and English Phrases
3. Context-free Grammars
4. Discussion

SYNTAX IS NOT MORPHOLOGY

Morphology deals with the internal structure of words

- syntax deals with combinations of words
- phrases and sentences

Morphology is often irregular

- syntax has its irregularities, but is usually regular
- syntax is mostly made up of general rules that apply across-the-board

SYNTAX IS NOT SEMANTICS

Semantics is about meaning; syntax is about structure alone

A sentence can be syntactically well-formed but semantically ill-formed:

- *Colorless green ideas sleep furiously*

Some well-known linguistic theories attempt to “read” semantic representations off of syntactic representations in a compositional fashion

(We’ll talk about these in a later lecture)

CONSTITUENCY AND ENGLISH PHRASES

CONSTITUENCY

One way of viewing the structure of a sentence is as
a *collection of nested constituents*

- **constituent:** a group of words that "go together"
  (or relate more closely to one another than to other words in the sentence)

Constituents larger than a single word are called *phrases*

Phrases can contain other phrases

NOUN PHRASES (NP)

The elephant arrived.

It arrived.

Elephants arrived.

The big ugly elephant arrived.

The elephant I love to hate arrived.

PREPOSITIONAL PHRASES (PP)

I arrived on Tuesday.

I arrived in March.

I arrived under the leaking rood.

Every **prepositional phrase** contains a **noun phrase.**

Remember how we defined prepositions?

SENTENCES OR CLAUSES (S)

- John loves Mary
- John loves the woman he thinks is Mary.
- Sometimes, John thinks he is Mary.
- It is patently false that sometimes John thinks he is Mary.

CONTEXT-FREE GRAMMARS

CONTEXT-FREE GRAMMARS

Vocabulary of terminal symbols: Σ

Set of non-terminal symbols: N

Special start symbols: $S \in N$

Production rules of the form $X \rightarrow \alpha$, where

$X \in N$

$\alpha \in (N \cup \Sigma)^*$

The grammars are called “context-free” because there is no context in the LHS of the rules — there is just one symbol.

NON-TERMINALS AND TERMINALS

A non-terminal symbol is on like S that can (and must!) be re-written as either:
- other non-terminal symbols
- terminal symbols

Non-terminals can be phrasal or pre-terminal (in which case they look like part-of-speech tags — Noun, Vern, etc.)

In natural language syntax, terminals are usually words

Non-terminals cannot be re-written; they mean you're done.

CONTEXT-FREE RULES

S → NP VP

NP → Det Noun

VP → Verb NP

Det → the, a

Noun → boy, girl, hotdogs

Verb → likes, hates, eats

CFG-S AS DECLARATIVE PROGRAMMING

One way to look at context-free grammars is as declarative programs
- think Prolog, SQL, or XQuery
- Instead of specifying how the task is to be accomplished…
    - how sentences are to be generated
    - how sentences are to be parsed
- … CFGs specify what is to be computed in terms of rules and let generalized computation mechanisms solve for the particular cases

The same goes for regular expressions as well as other types of grammars.

BUILDING NOUN PHRASES

NP → Det NounBar

NP → ProperNoun

NounBar → Noun

NounBar → AP NounBar

NounBar → NounBar PP

AP → Adj AP

AP → Adj

PP → Preposition NP

TERMINOLOGY

Grammatical: said of a sentence *in* the language

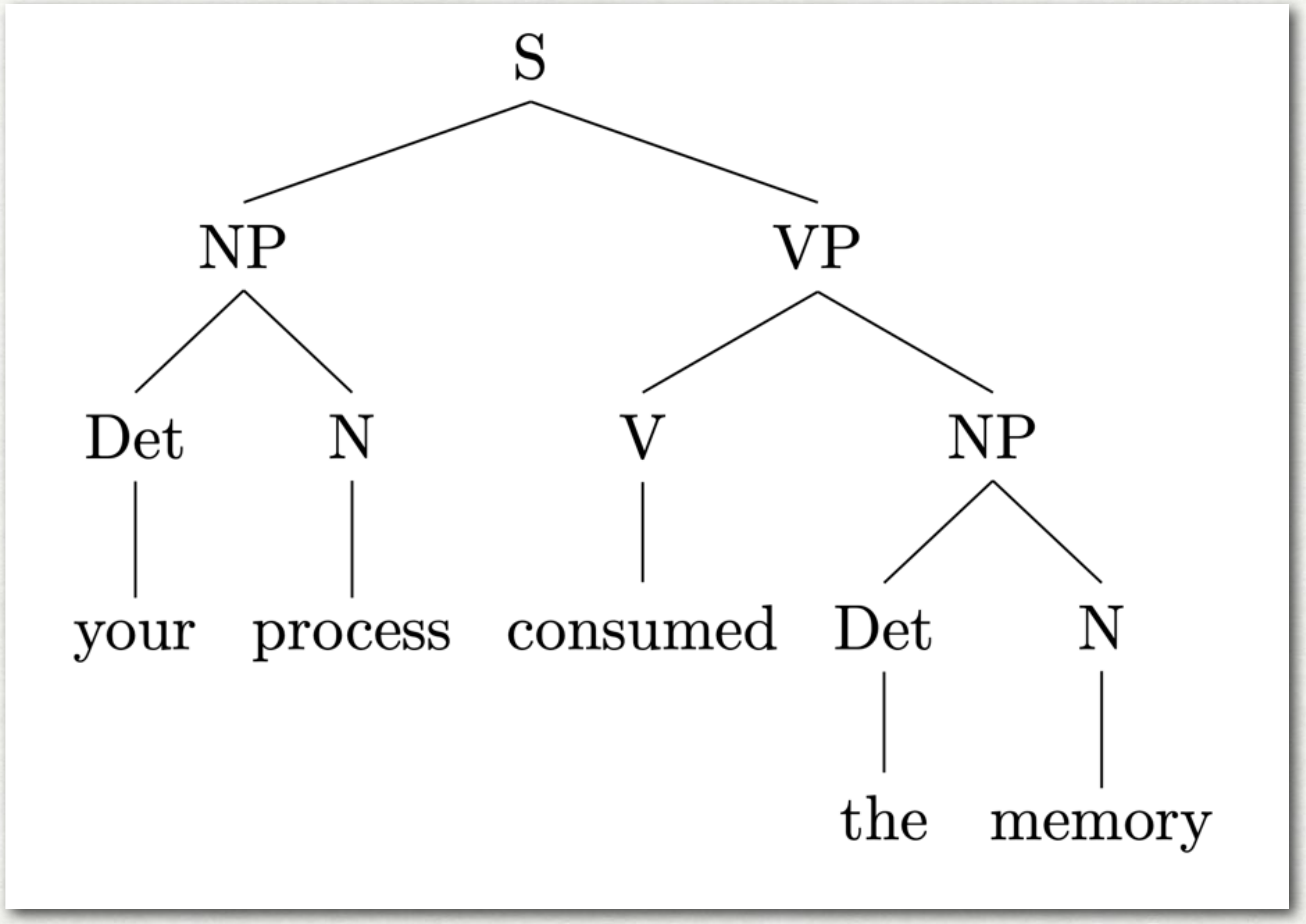
Ungrammatical: said of a sentence *not in* the language

Derivation: sequence of top-down production steps

Parse Tree: graphical representation of the derivation

A string is grammatical iff there exists a derivation for it.

A (CONSTITUENCY) PARSE TREE



AMBIGUITY

S → NP VP

NP → Det Noun

VP → Verb NP

VP → Verb PP

PP → Prep NP

Det → the, a

Noun → boy, girl, hotdogs, park

Verb → likes, hates, eats, sees

Prep → in, with

GRAMMATICALITY – IT VARIES

I’ll write the company

I’ll write to the company

It needs to be washed

It needs washed

They met Friday to discuss it

They met on Friday to discuss it

ON GETTING IT RIGHT

CFGs provide you with a tool set for creating grammars
- grammars that work well (for a given application)
- grammars that work poorly (for a given application)

There is nothing about the theory of CFGs that tells you, a priori, what a "correct" grammar for a given application looks like

A good grammar is generally one that:
- doesn't over-generate too much (high precision)
- doesn't under-generate too much (high recall)

What these look like in practice is going to vary with your application space.

MOTIVATION

WHY ARE WE BUILDING GRAMMARS?

Consider the following:

- Oswald shot Kennedy
- Kennedy was shot by Oswald
- Oswald was shot by Ruby

Now answer:

- Who shot Kennedy?
- Who shot Oswald?

WHY ARE WE BUILDING GRAMMARS?

Active/Passive

- Oswald shot Kennedy
- Kennedy was shot by Oswald

Relative clauses

- Oswald who shot Kennedy was shot by Ruby
- Kennedy who Oswald shot didn’t shoot anybody

WHO DID WHAT TO WHOM

There are multiple reasons to build grammars but one important reason is *knowing who did what to whom*

A parse tree does not tell us this directly, but it is one step in the process of discovering grammatical relations (subject, object, etc…) which can help us discover semantic roles (agent, patient, etc)

LANGUAGE MYTHS: SUBJECT

Myth 1: the subject is the first noun phrase in a sentence

Myth 2: the subject is the actor in a sentence

Myth 3: the subject is what the sentence is about

All of these are **often** true, but none of them is **always** true, or tells you what a subject really is (or how to use it in NLP).

SUBJECT, OBJECT, AND DEPENDENCIES

SUBJECT AND OBJECT

Syntactic (not semantic)

- The batter hit the ball. [subject is the semantic agent]

- The ball was hit by the batter. [subject is the semantic patient]

- The ball was given a whack by the batter. [subject is the semantic recipient]

- {George, the key, the wind} opened the door.

Subject IS DIFFERENT FROM topic

- I just married the most beautiful woman in the world.
- Now beans, I like.
- As for democracy, I think it's the best form of government.

SUBJECT AND OBJECT

English subjects

- agree with the verb
- when pronouns, in nominative case (I/she/he vs me/her/him)
- omitted from infinitive clauses (I tried __ to read the book, I hoped __ to be chosen)

English objects

- When pronouns, in accusative case
- Become subjects in passive sentences

DEPENDENCY GRAMMAR

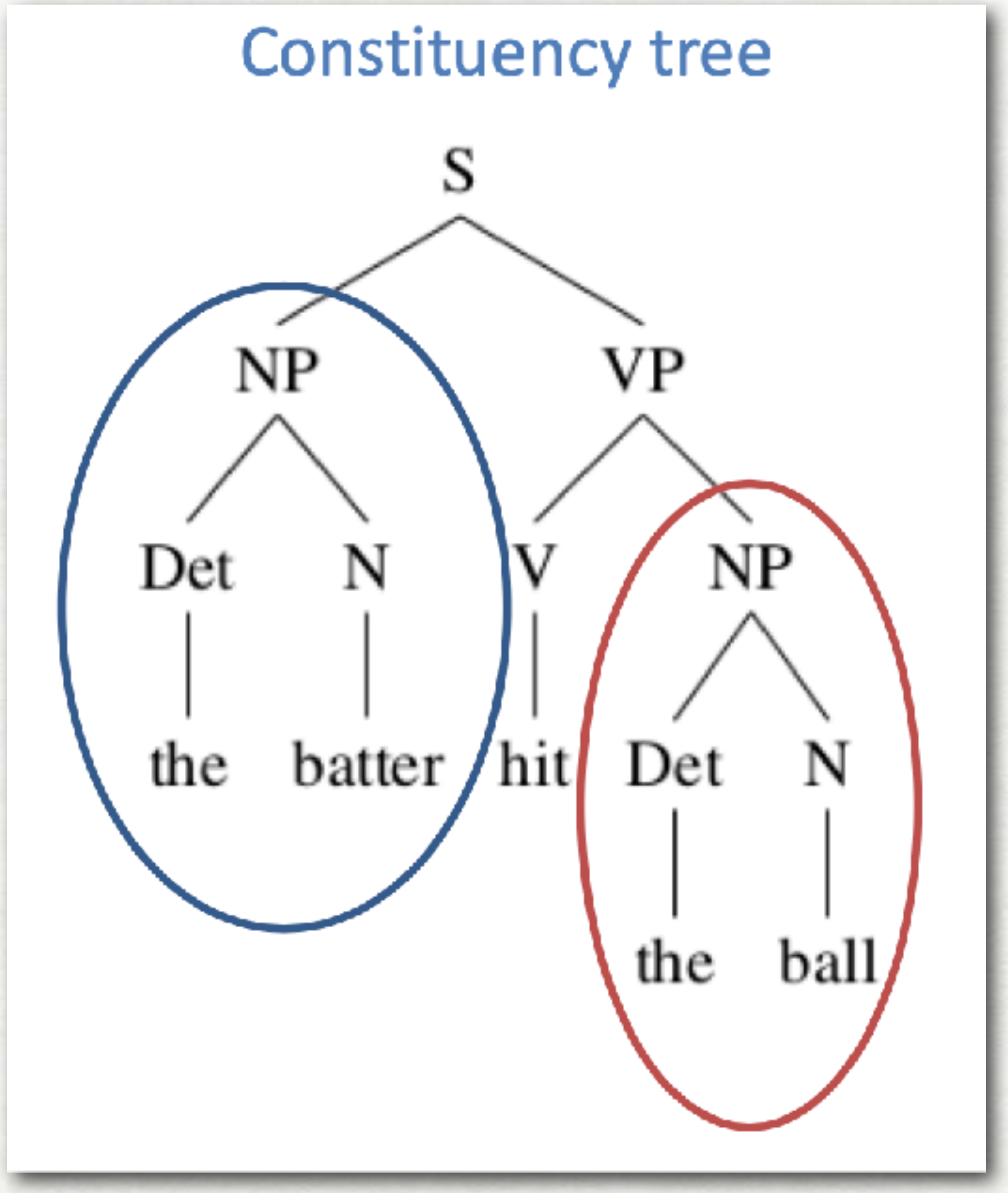
There is another way of looking at syntax that highlights relations like *subject* and *object*

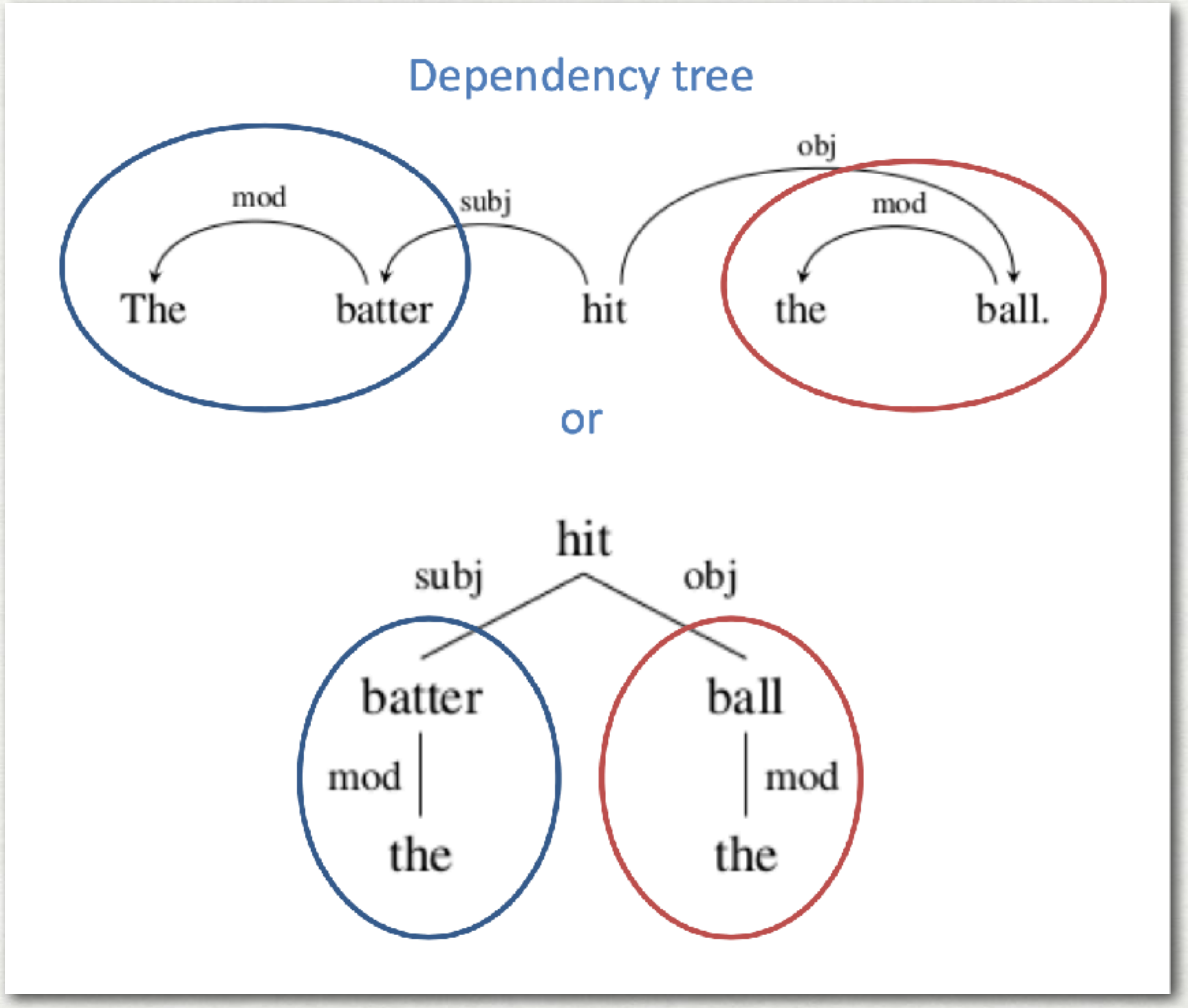
Dependency grammar

- Bilexical dependencies
    - relationships between two words
    - one is "head" and one is "dependent"
    - labels like "subj" and "obj" on arcs

Example: verbs are **heads** relative to their subject and objects, which are **dependents**

DEPENDENCIES





ADVANTAGES AND DISADVANTAGES

Advantages of Constituency/Phrase Structure Grammars

- There are widely agreed-upon tests for constituency; there is little agreement on what constitutes a dependency relation

- Constituency maps more cleanly on to formal semantic representations than dependency

- This makes constituency useful in natural language understanding.

Advantages of Dependency Grammars

- It is easier to identify grammatical relations (like subject and object) in a dependency parse

- Dependency parses of sentences having the same meaning are more similar across languages than constituency parses

- Dependency parses are also useful for NLP (ask Google)

- Dependency trees are typically simpler.

ADDITIONAL NOTES

Some approaches to syntax, including **Lexical Functional Grammar** or **LFG**, use dependency and constituency as parallel representations

Stanford parser does both constituency and dependency parsing (neural network dep. parser)

Many other parsers for both constituency and dependency exist:

- Berkeley Parser,
- MaltParser
- SyntaxNet & Parsey McParseface
- TurboParser
- UDPipe and UDify

NEXT CLASS

We will talk a lot about constituency parsing and CFGs

CFGs may not be entirely adequate for capturing the syntax of natural languages

- they are almost adequate
- they are computationally well-behaved (you can build relatively efficient parsers for them)
- but they are not very convenient as a means for hand-crafting a grammar
- Also, they are not probabilistic

We will revisit these properties of CFGs