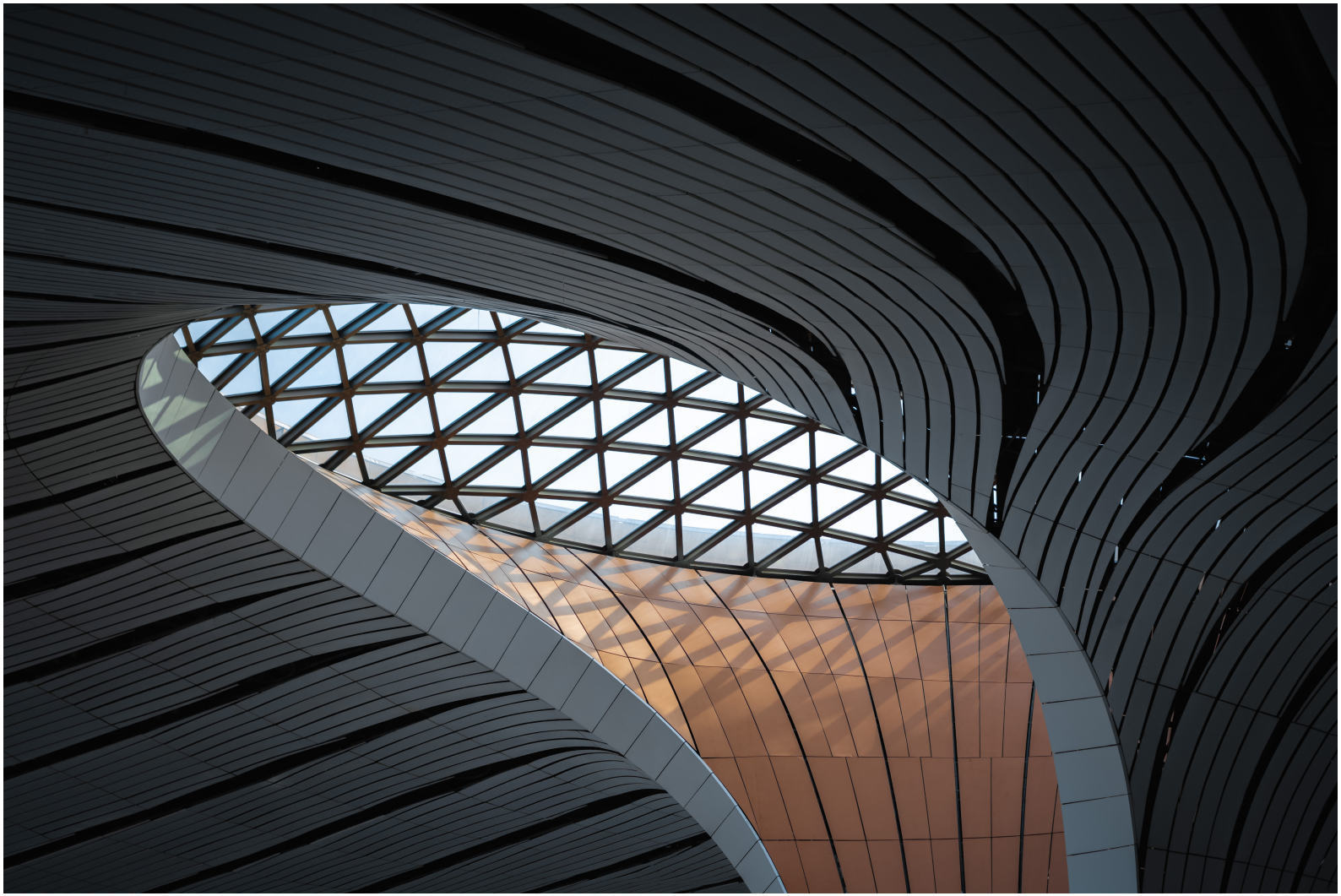
Natural Language Processing

Lecture 8 : Phrase-Structure Parsing (part I)

Master Degree in Computer Engineering
University of Padua
Lecturer : Giorgio Satta

Lecture partially based on material originally developed by :
Mark-Jan Nederhof, University of St. Andrews

Syntax



Rudy Dong from Unsplash

Syntax

In linguistics, **syntax** is the set of rules, principles, and processes that govern the structure of sentences in a given language, including word order.

The term syntax is also used to refer to the study of such principles and processes, in a way that generalises across languages.

We need syntactic analysis in order to be able to perform several NLP tasks.

The same also holds for compiling of programming languages, where this methodology is called syntax-directed translation.

Constituents

In syntactic analysis a **constituent**, also called **phrase**, is a group of words that function as a single unit within a hierarchical structure.

Example : The sentence "He saw the house on the hill" has the following constituents (among others)

- He
- the house
- on the hill

while the following sequences are **not** constituents

- He saw the
- house on
- on the

Constituents

The constituent structure of a sentence is identified using **constituency tests**.

Several tests, we only present few of them.

General substitution: Replaces the test phrase with some other phrase of the same type

Example :

- Drunks could put off **the customers**
- Drunks could put off **our guests**

Constituents

Coordination: One can compose several constituents together using coordination

Example : He **was taking lessons** and **is now teaching himself**

Topicalization: If a sequence can as a whole move to a different location without affecting grammaticality

Example :

- He is taking classes **to improve his English**
- **To improve his English**, he is taking classes

Constituents

There are several types of constituents, each characterised by:

- The context in which they can occur
- Their internal structure

Sentence, abbreviated S, is a constituent representing a complete proposition or clause.

Example :

- **This is a sentence**
- Alice mentioned **that Bob has been to Venice**

Constituents

Noun phrase, abbreviated NP:

- Typically built around a common noun, proper noun, or personal pronoun
- Can occur just before a verb as **subject**, or after a verb as **object**

Example :

- **I** prefer **the morning flight**
- **My colleagues** prefer **the flight in the morning**

Constituents

Verb phrase, abbreviated VP:

- Typically starts with at least one verb
- Optionally followed by some argument, such as NP or S

Example :

- A man **put the money quickly in the box**
- Alice **mentioned that Bob has been to Venice**

Constituents

Prepositional phrase, abbreviated PP:

- Typically starts with preposition followed by a NP
- May be omitted from a sentence

Example :

- I want to eat sushi **with chopsticks**
- I want to eat soup **with onions**

In first case the PP is part of a VP, in the second case is part of a NP.

Constituents

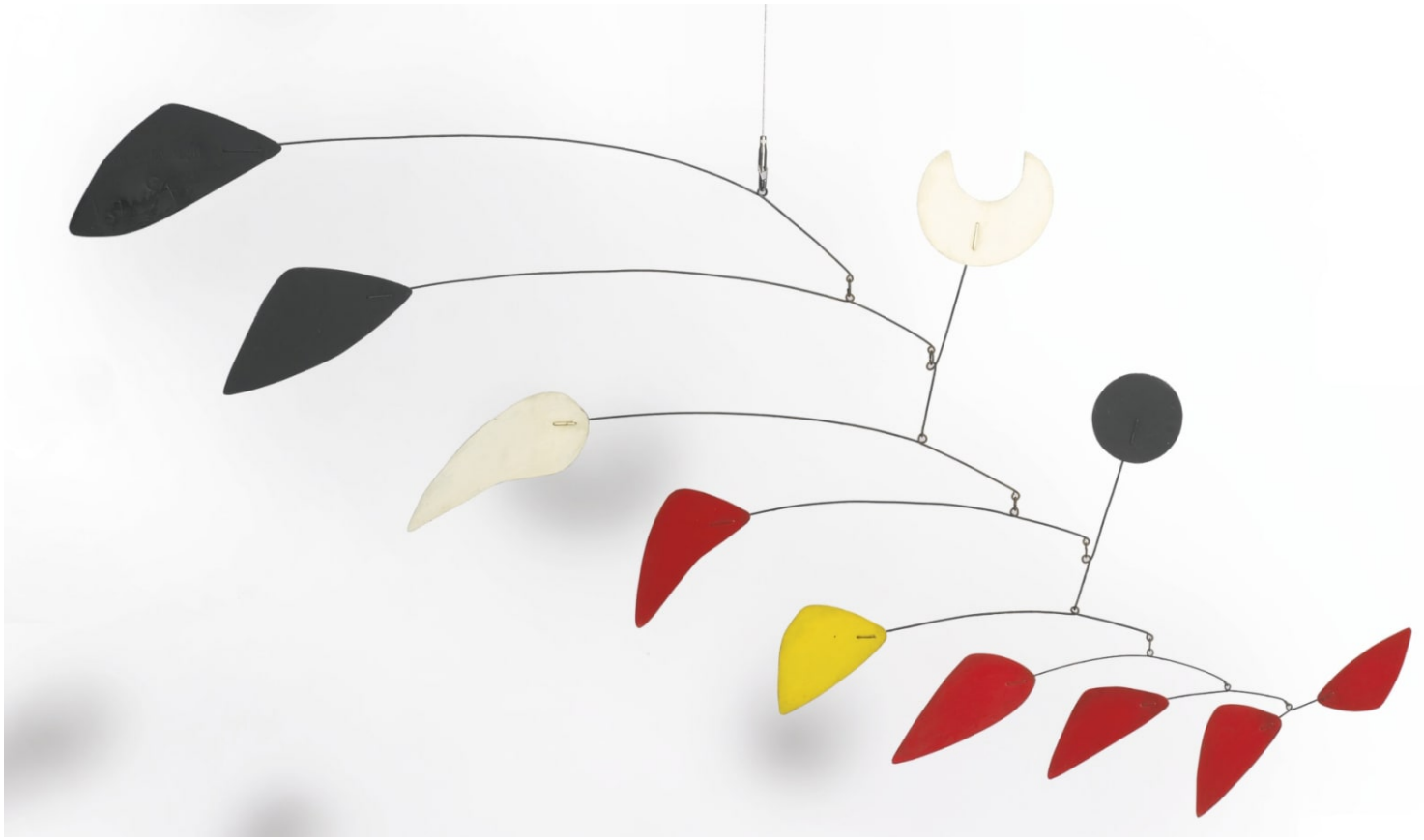
Adjective phrase, abbreviated AP:

- Typically built around some adjective
- Possibly preceded by adverbs

Example :

- Give me a **cheap** fare
- Give me the **least expensive** fare

Phrase structure



Calder Foundation, New York

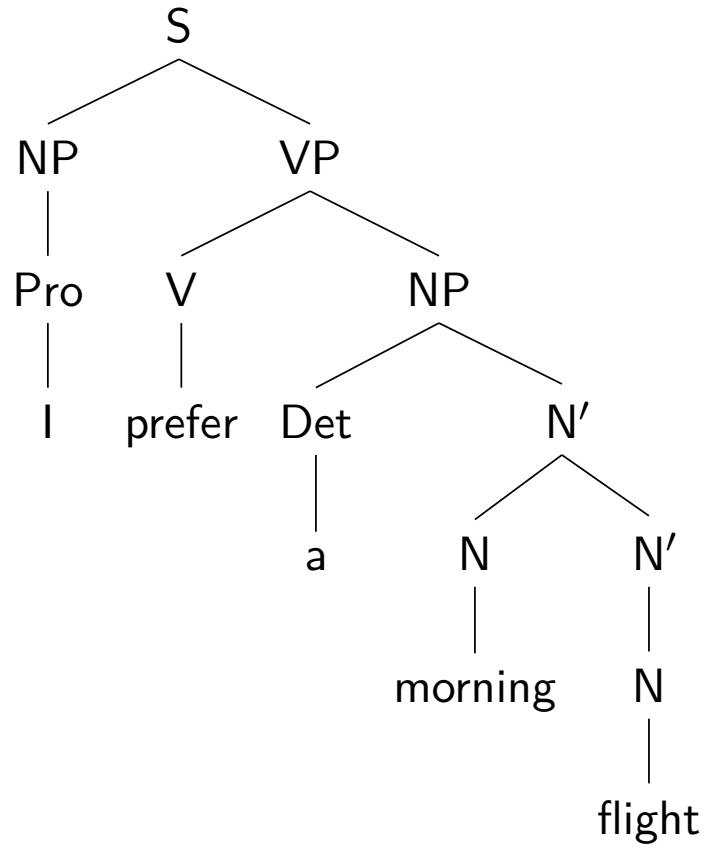
Phrase structure

Phrase structures are tree-like representations used to describe a given language's syntax on the basis of

- the linear order of words in the sentence
- the grouping of words into constituents (phrases)
- the hierarchical relation between constituents

Phrase structure

Example : Phrase structure tree for sentence "I prefer a morning flight"



Using X-bar theory (Jakendoff, 1974) to represent phrase internal structure.

Phrase structure

We can also represent a parse tree in a linear format called **bracketed** notation

Example :

[$_S$ [$_{NP}$ [$_{Pro}$ I]]
[$_{VP}$ [$_V$ prefer]
[$_{NP}$ [$_{Det}$ a] [$_{N'}$ [$_N$ morning] [$_{N'}$ [$_N$ flight]]]]]]

Sometimes, only a subset of interest of all the phrases is indicated.

Example :

I [$_{VP}$ prefer [$_{NP}$ a morning [$_N$ flight]]]

Head

The **head** is the word in the phrase that is grammatically the most important.

The head identifies the phrase type: N is the head of an NP, V is the head of a VP, and so forth.

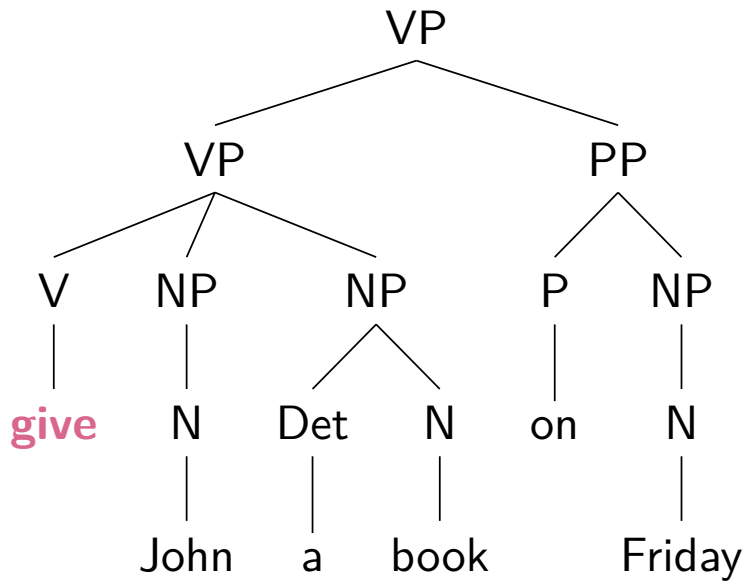
The head selects its **complements** (sub-phrases), classified as:

- **arguments**: inherent to the meaning of the phrase; they appear in fixed number depending on the head's semantics
- **modifiers**: optional phrases which merely supplement the head with additional information; they can appear in any number

Thus each phrase is built around its head: we say that the head **projects** the entire phrase structure.

Head

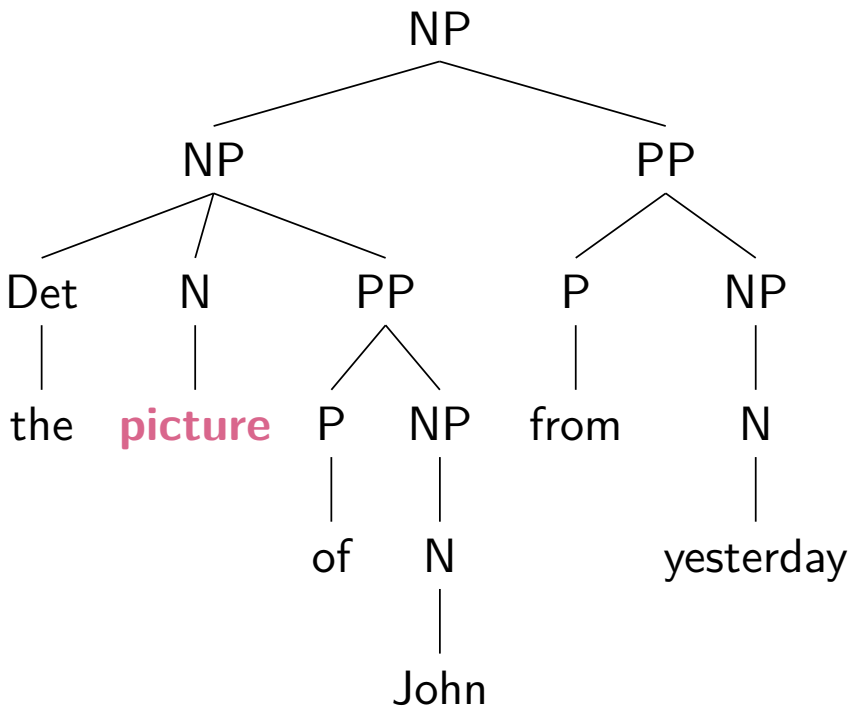
Example : Internal structure of a VP, as projected by the head V



Arguments: John, a book; modifier: on Friday.

Head

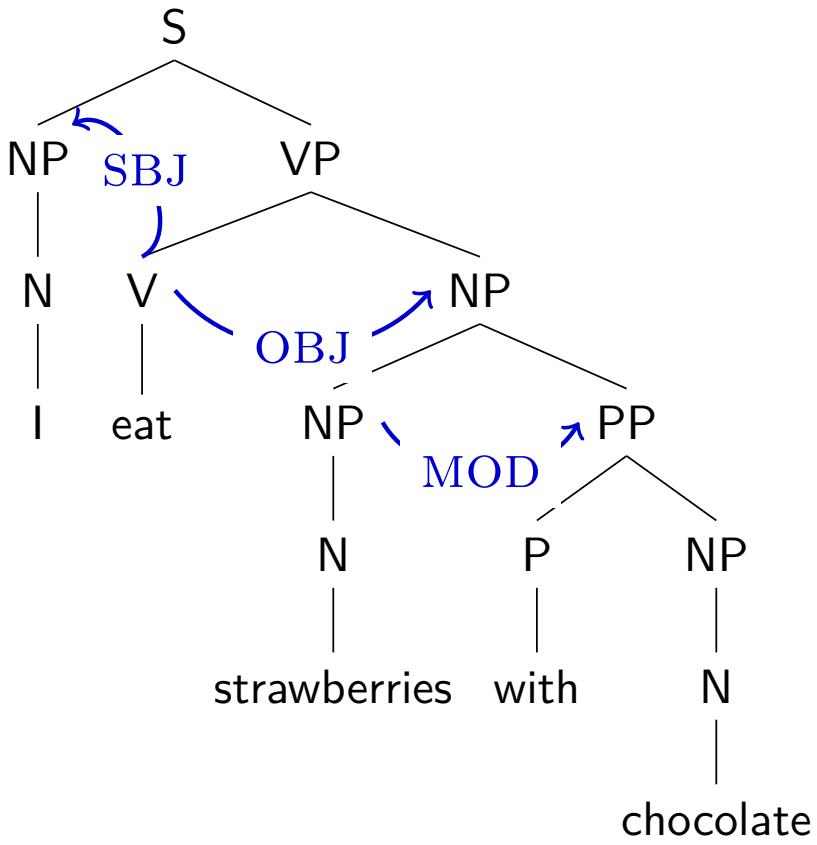
Example : Internal structure of an NP, as projected by the head N



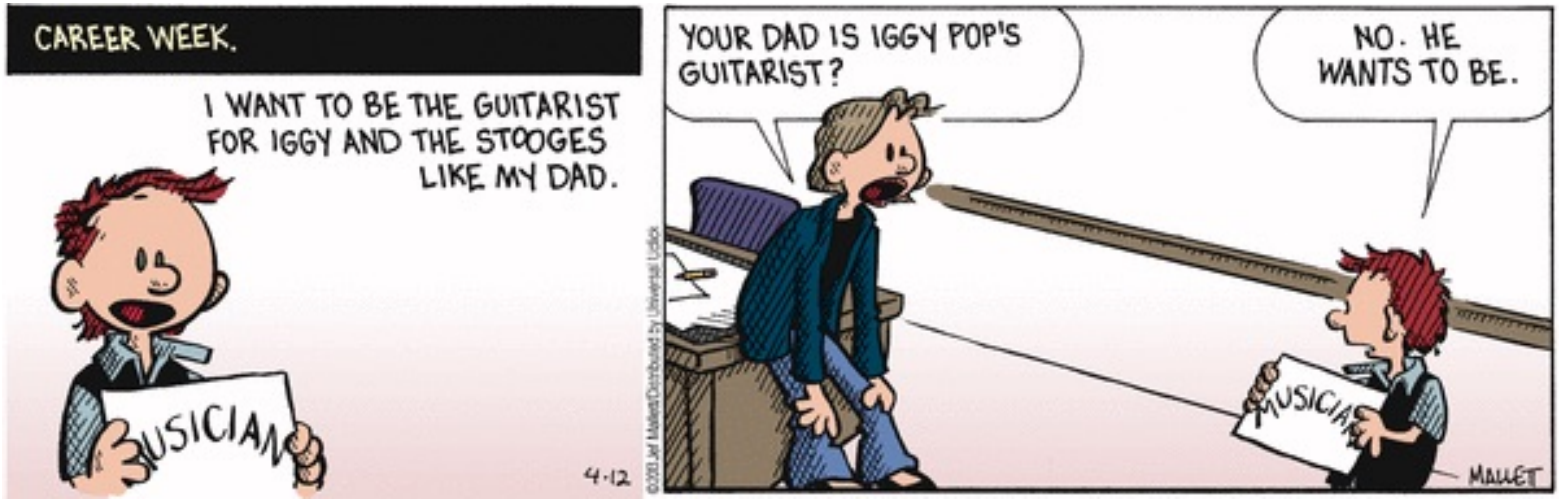
Argument: of John; modifier: from yesterday.

Grammatical relations

Grammatical relations such as subject, direct object, and modifier can be retrieved from phrase structure.



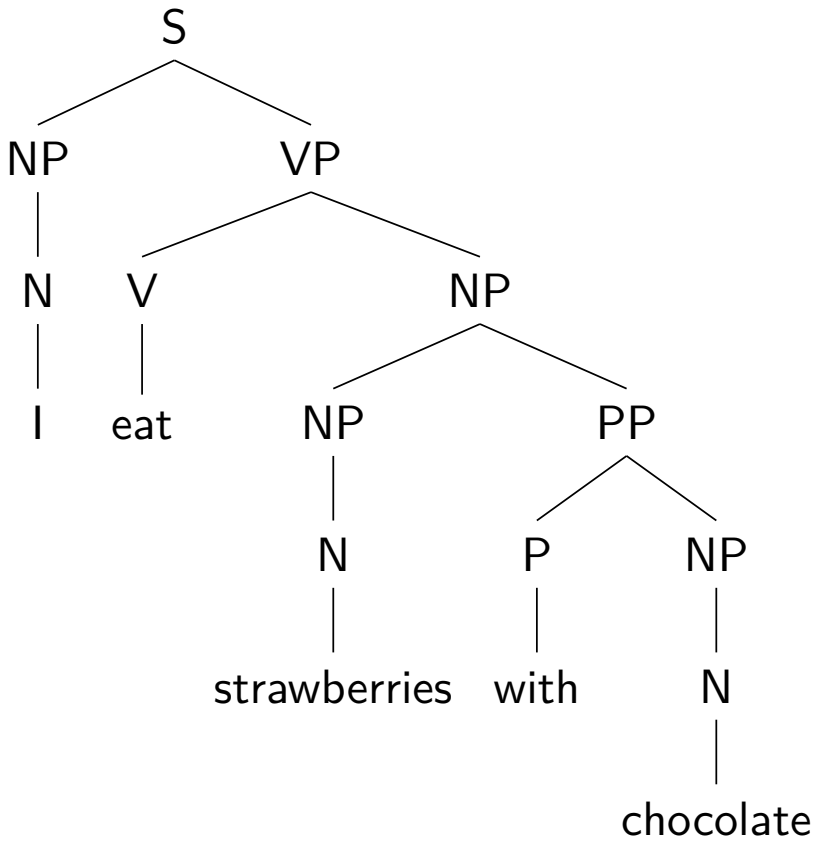
PP attachment



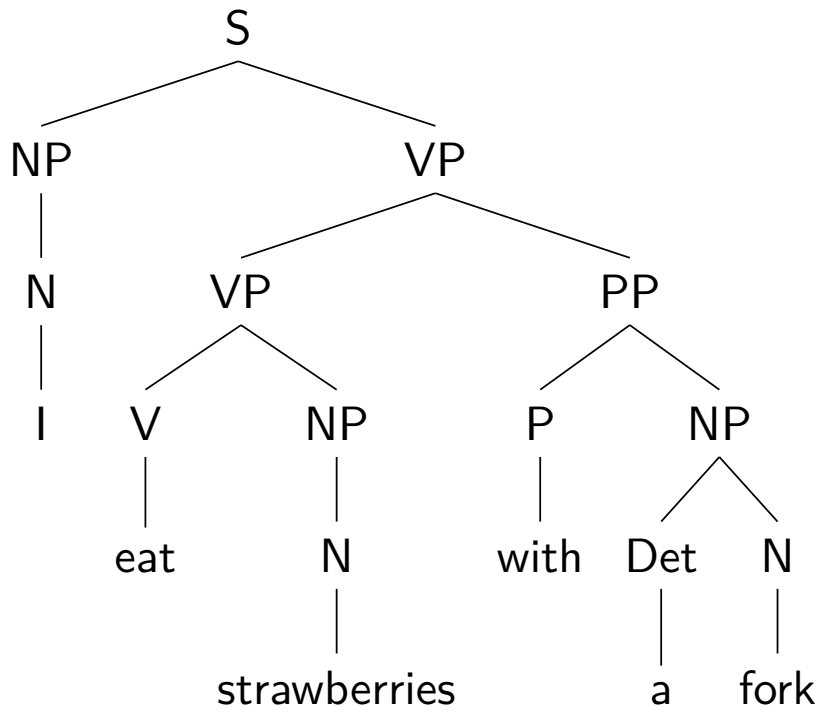
https://languagelog.ldc.upenn.edu/nll/?p=4566

PP attachment

PP attachment is an important problem in syntactic parsing that affects the interpretation of the sentence.

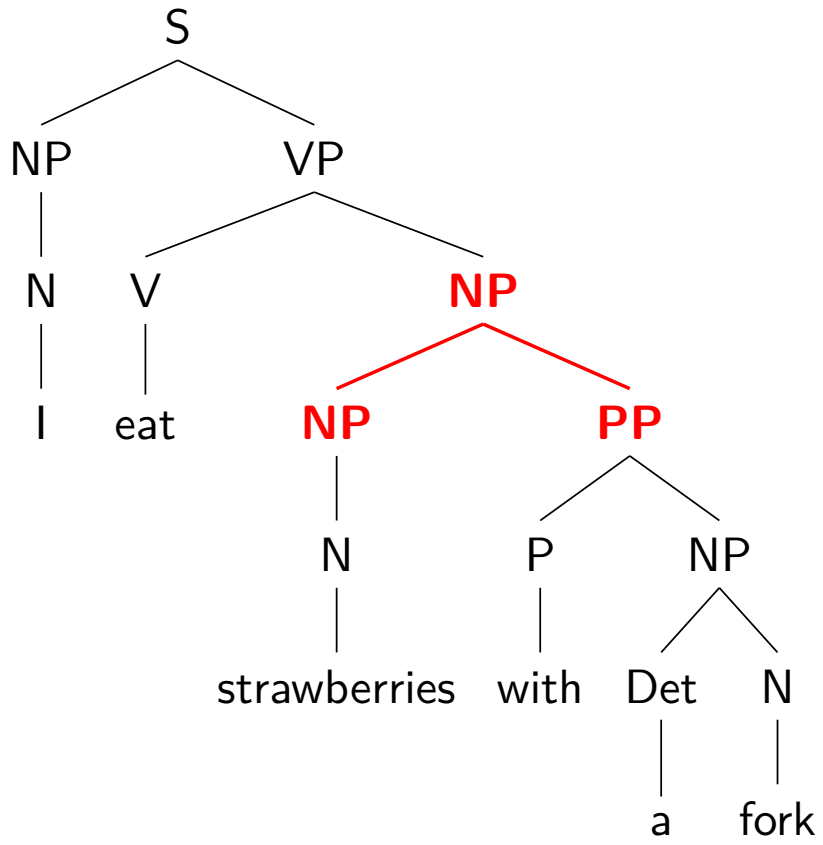


PP attachment



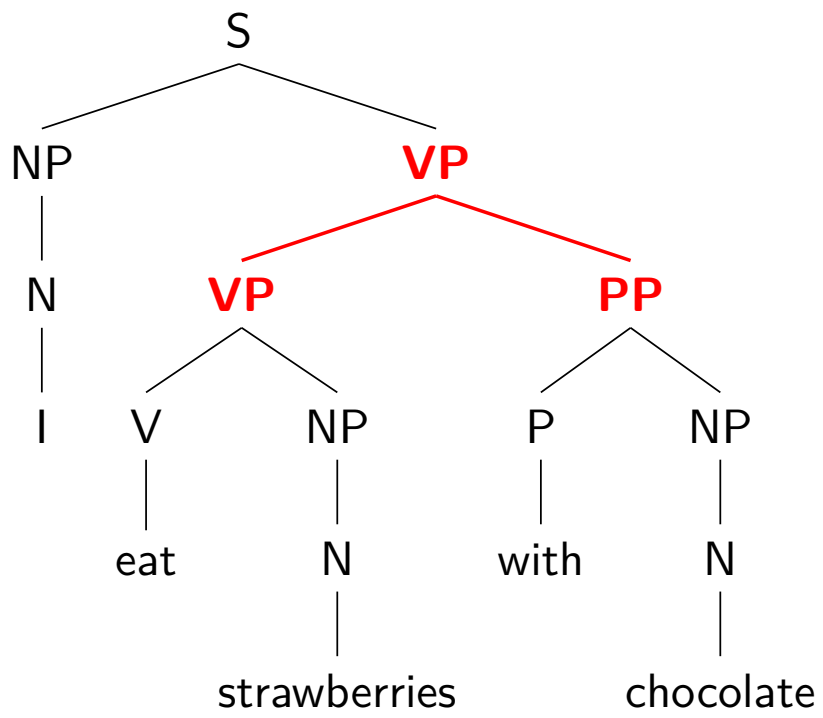
PP attachment

Wrong PP attachment marked in red, leading to wrong semantic interpretation of the sentence.



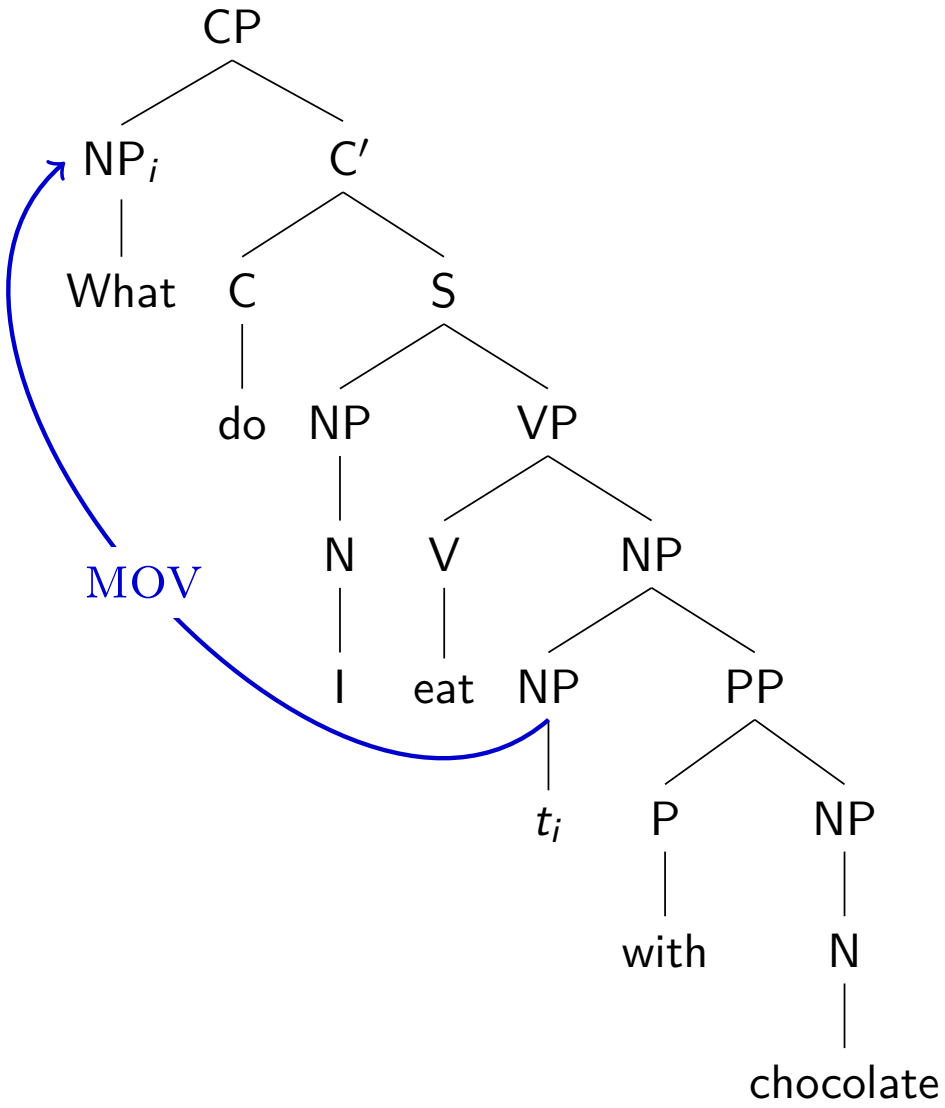
PP attachment

Wrong PP attachment marked in red, leading to wrong semantic interpretation of the sentence.



Wh-movement

Long distance syntactic movement, or wh-movement, can be represented in phrase structure by means of so-called **traces**.



Treebanks

Treebanks

A phrase structure **treebank** is a parsed text corpus that annotates the syntactic structure of each sentence, **resolving ambiguity**.

Treebanks are used to train phrase structure grammars, that is, grammars that can model phrase structures.

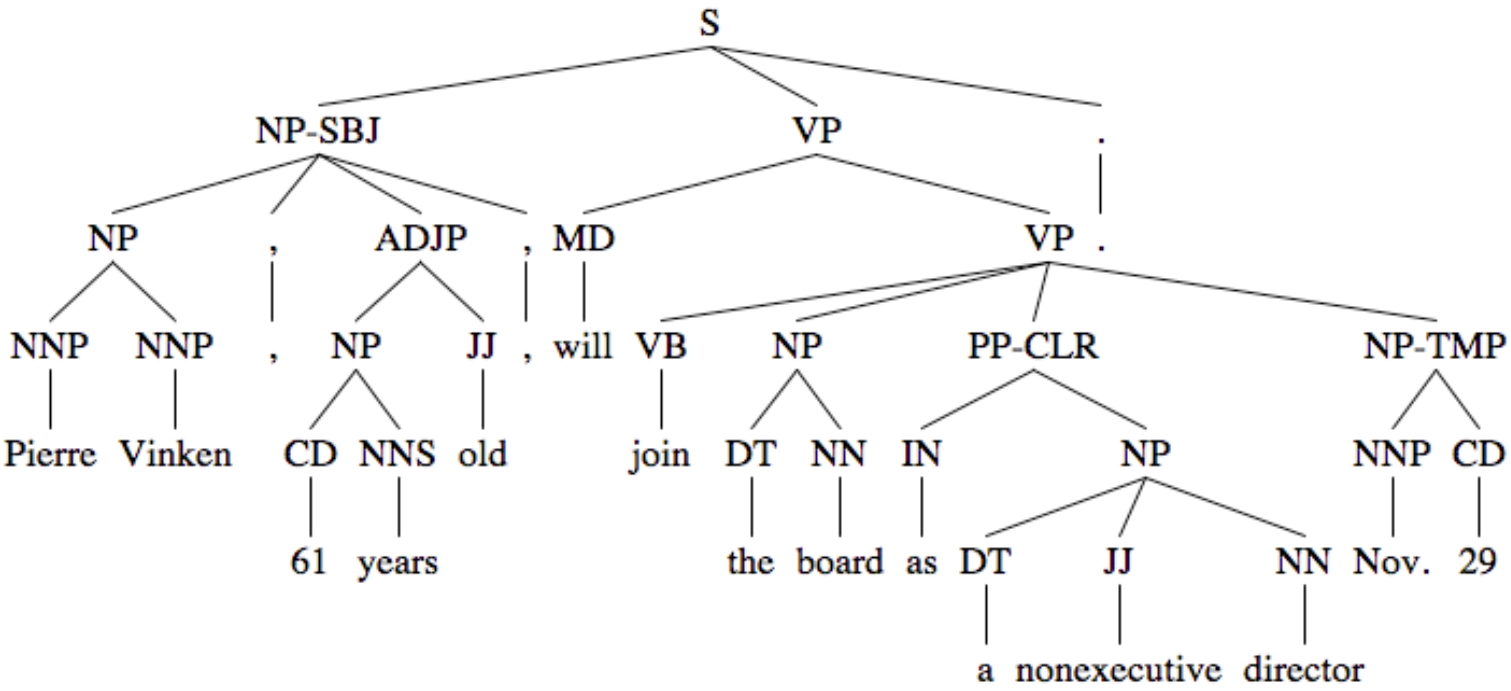
Context-free grammars are an example of phrase structure grammars.

The **Penn Treebank** was the first large-scale treebank. Published in the early 1990s, it revolutionised NLP.

Produced from the Brown, Switchboard, ATIS, and Wall Street Journal English corpora.

Several phrase structure treebanks publicly available for many languages.

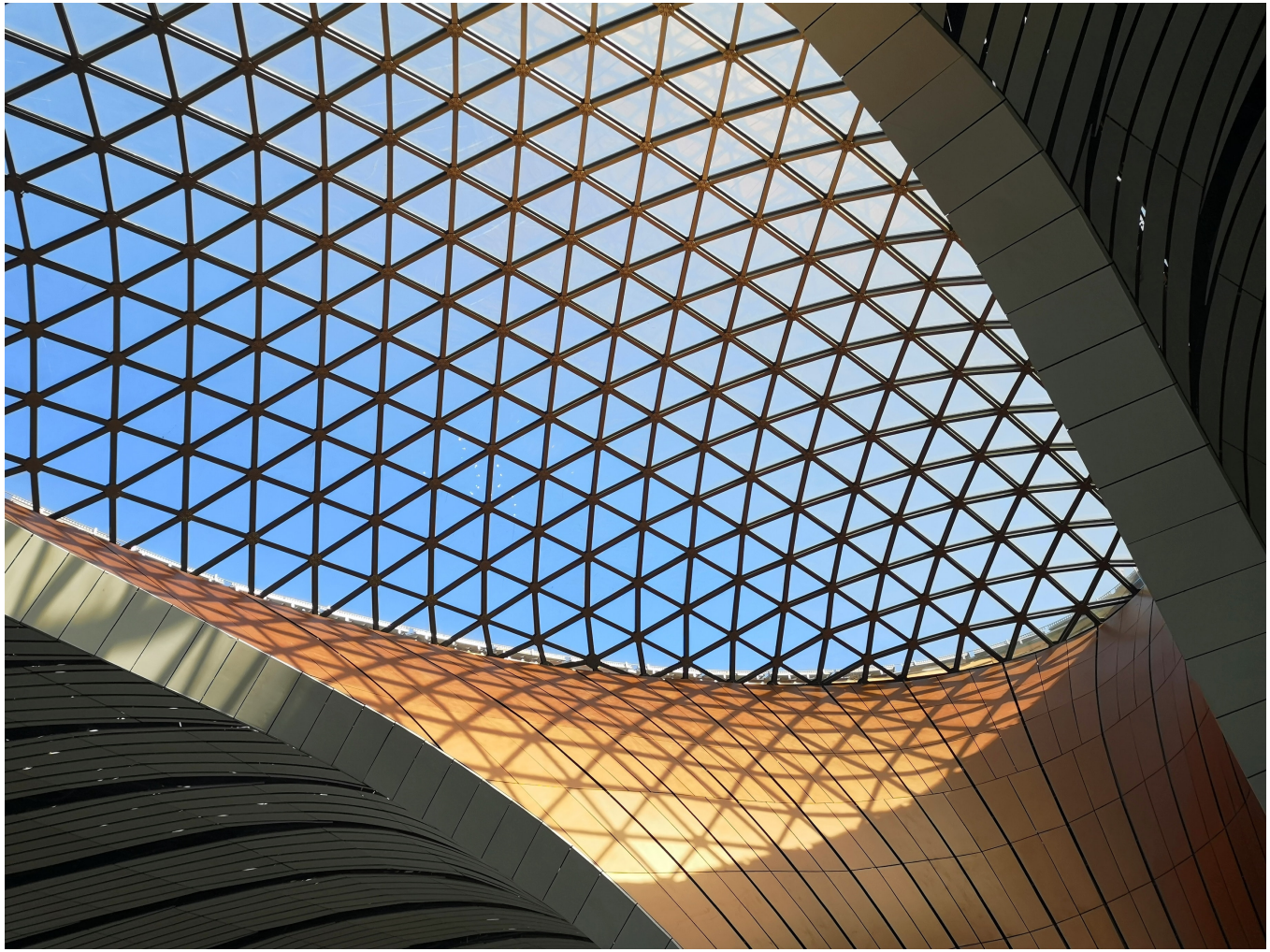
Treebanks


S
NP-SBJ
VP
.
NP
,
ADJP
,
MD
VP
.
NNP
NNP
,
NP
JJ
,
will
VB
NP
PP-CLR
NP-TMP
Pierre
Vinken
CD
NNS
old
join
DT
NN
IN
NP
NNP
CD
61
years
the
board
as
DT
JJ
NN
Nov.
29
a
nonexecutive
director

Treebanks

```
( (S
    (NP-SBJ
      (NP (NNP Pierre) (NNP Vinken) )
      (\, \,)
      (ADJP
        (NP (CD 61) (NNS years) )
        (JJ old) )
      (\, \,) )
    (VP (MD will)
      (VP (VB join)
        (NP (DT the) (NN board) )
        (PP-CLR (IN as)
          (NP (DT a) (JJ nonexecutive) (NN director) ))
        (NP-TMP (NNP Nov.) (CD 29) )))
    (\. \.) ))
( (S
    (NP-SBJ (NNP Mr.) (NNP Vinken) )
    (VP (VBZ is)
      (NP-PRD
        (NP (NN chairman) )
        (PP (IN of)
          (NP
            (NP (NNP Elsevier) (NNP N.V.) )
            (\, \,)
            (NP (DT the) (NNP Dutch) (VBG publishing) (NN group) )))))
    (\. \.) ))
```

Context-free grammars



Katie Yang from Unsplash

Context-free grammars

Context-free grammars (CFGs) are the backbone of many formal models of the syntax of natural language.

We will also consider an alternative model in the next lecture.

Probabilistic CFGs allow to define probability distributions over strings and trees generated by a CFG, which can be used to solve ambiguity.

Lexicalized CFGs allow to define expressive models which account for lexical selection.

Efficient algorithms exist for parsing sentences based on CFGs, using dynamic programming techniques.

Context-free grammars

Recap:

- nonterminal and terminal symbols; start symbol S
- rules of the form $A \rightarrow \alpha$
- notion of rewriting step, expressed by relation $\Rightarrow$
- notion of derivation, expressed by relation $\overset{*}{\Rightarrow}$, and notion of generated language
- notion of derivation tree

Context-free grammars

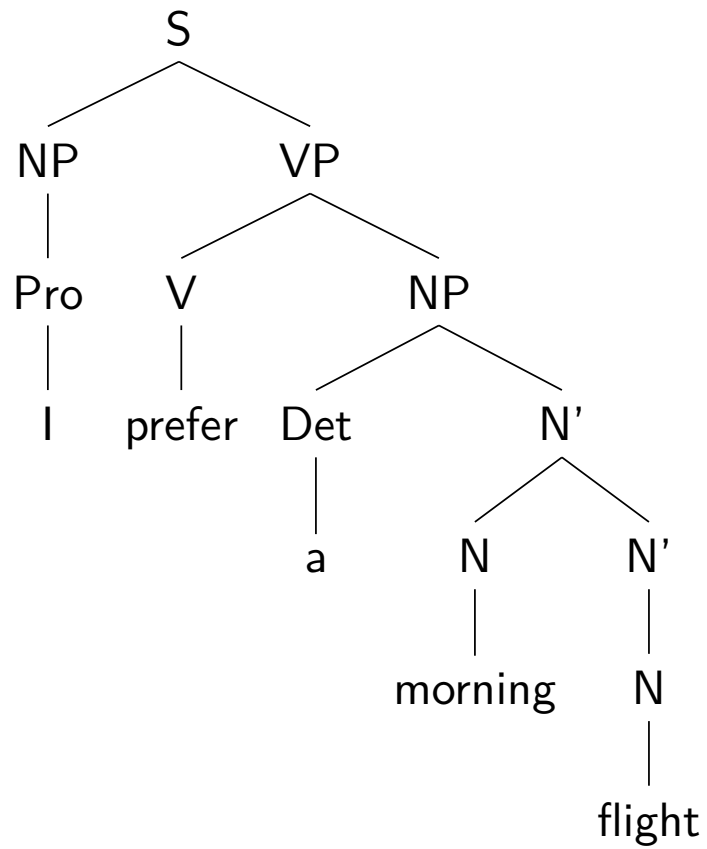
Example : Very simple CFG rules for a fragment of English

S	→	NP VP	Pro	→	I
VP	→	V NP	V	→	prefer
NP	→	Det Nom	Det	→	a
NP	→	Pro	N	→	morning
N'	→	N N'	N	→	flight
N'	→	N			

Symbols V, N, Pro, Det are also called **preterminals**; these symbols serve as POS tags.

Context-free grammars

Generated phrase structure tree.



Context-free grammars

A CFG can be immediately read from a phrase structure treebank.

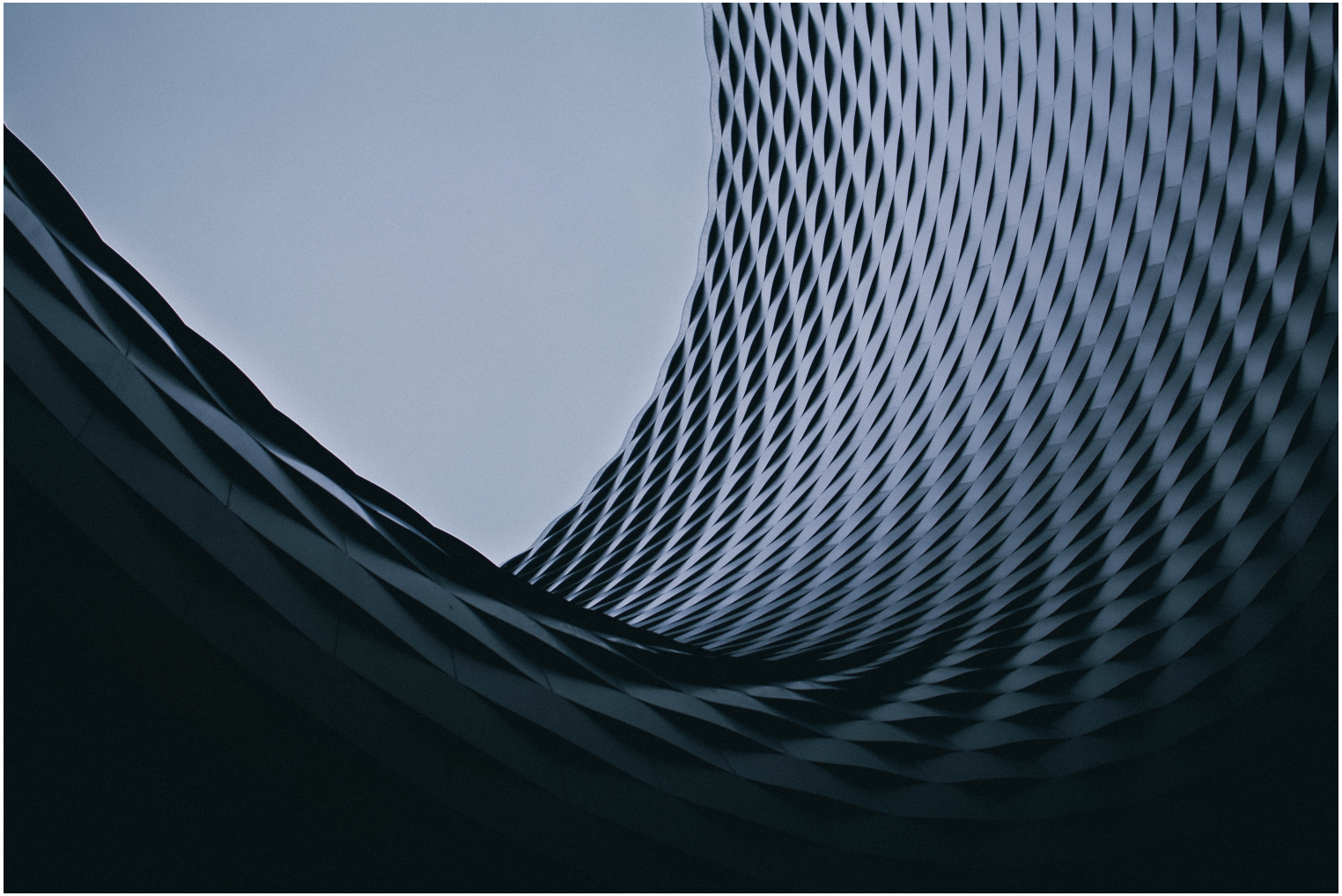
The grammar underlying the Penn Treebank is relatively **flat**, resulting in very many and very long rules.

Example : VP rules accounting for PP sequences of various length and various possible arrangement of PP and ADVP arguments

VP	→	VBD PP	VP	→	VB ADVP PP
VP	→	VBD PP PP	VP	→	VB PP ADVP
VP	→	VBD PP PP PP	VP	→	ADVP VB PP
VP	→	VBD PP PP PP PP			

Approximately 4,500 different rules for VPs.

Probabilistic CFG



Victor Iaqu from Unsplash

Probabilistic CFG

A **probabilistic context-free grammar** (PCFG) allows to define a probability distribution over the set of generated parse trees.

This is done by assigning probabilities to rules, and by multiplying rule probabilities to compute the probability of a parse tree.

Under this view, rule applications are independent events.

Suppose a sentence has several parse trees. If rule probabilities are **estimated** appropriately, we could solve ambiguity by selecting the parse tree with the highest probability.

Probabilistic CFG

For each rule $A \rightarrow \alpha$, we specify the probability that the rule applies to A to produce α.

For historical reasons, this is written $P(A \rightarrow \alpha)$. However, it would be more correct to use the notation $P(A \rightarrow \alpha \mid A)$ or $P(\alpha \mid A)$, as we are conditioning on A.

A PCFG is **proper** if for each A we have

$$\sum_{\alpha} P(A \rightarrow \alpha) = 1$$

Note that for each A and for all possible right-hand sides α, $P(A \rightarrow \alpha)$ is a probability distribution.

Probabilistic CFG

Example : The following is a fragment of a proper PCFG

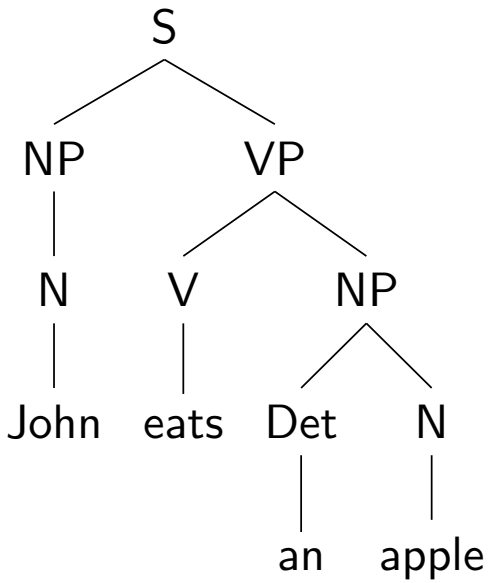
S	→	NP VP	.80	NP	→	Pron	0.35
S	→	Aux NP VP	.15	NP	→	NN	0.30
S	→	VP	.05	NP	→	Det N'	0.20
				NP	→	N'	0.15

Probabilistic CFG

The probability of a parse tree is computed by multiplying the probabilities of all rule occurrences.

Thus, with every additional rule, the probability decreases.

Example :



Tree probability:

$$
\begin{aligned}
&P(\mathrm{S} \rightarrow \mathrm{NP\ VP}) \times \\
&P(\mathrm{NP} \rightarrow \mathrm{N}) \times \\
&P(\mathrm{N} \rightarrow \mathrm{John}) \times \\
&P(\mathrm{VP} \rightarrow \mathrm{V\ NP}) \times \\
&P(\mathrm{V} \rightarrow \mathrm{eats}) \times \\
&P(\mathrm{NP} \rightarrow \mathrm{Det\ N}) \times \\
&P(\mathrm{Det} \rightarrow \mathrm{an}) \times \\
&P(\mathrm{N} \rightarrow \mathrm{apple})
\end{aligned}
$$

Probabilistic CFG

For a rule $A \rightarrow \alpha$ and a tree t, let $f(A \rightarrow \alpha, t)$ be the number of times $A \rightarrow \alpha$ is used in t.

Formally, the probability of t is defined as:

$$P(t) = \prod_{A \rightarrow \alpha} (P(A \rightarrow \alpha))^{f(t, A \rightarrow \alpha)}$$

Probabilistic CFG

The probability of a string w generated by the grammar is the sum of the probabilities of all parse trees of w.

Formally, let $T(w)$ be the set of all parse trees of w. Then

$$P(w) = \sum_{t \in T(w)} P(t)$$

A PCFG is **consistent** if $\sum_w P(w) = 1$. In words: the PCFG describes a probability distribution over the generated language.

Recall that the generated language is the set of finite sequences (strings) derived from the start symbol.

Consistency also means $\sum_t P(t) = 1$ for parse trees t generated by the grammar.

Probabilistic CFG

Surprisingly enough, a proper PCFG is not always consistent.

Example : Consider the following proper PCFG:

$$\begin{array}{lcll} S & \rightarrow & S\ S & 0.9 \\ S & \rightarrow & a & 0.1 \end{array}$$

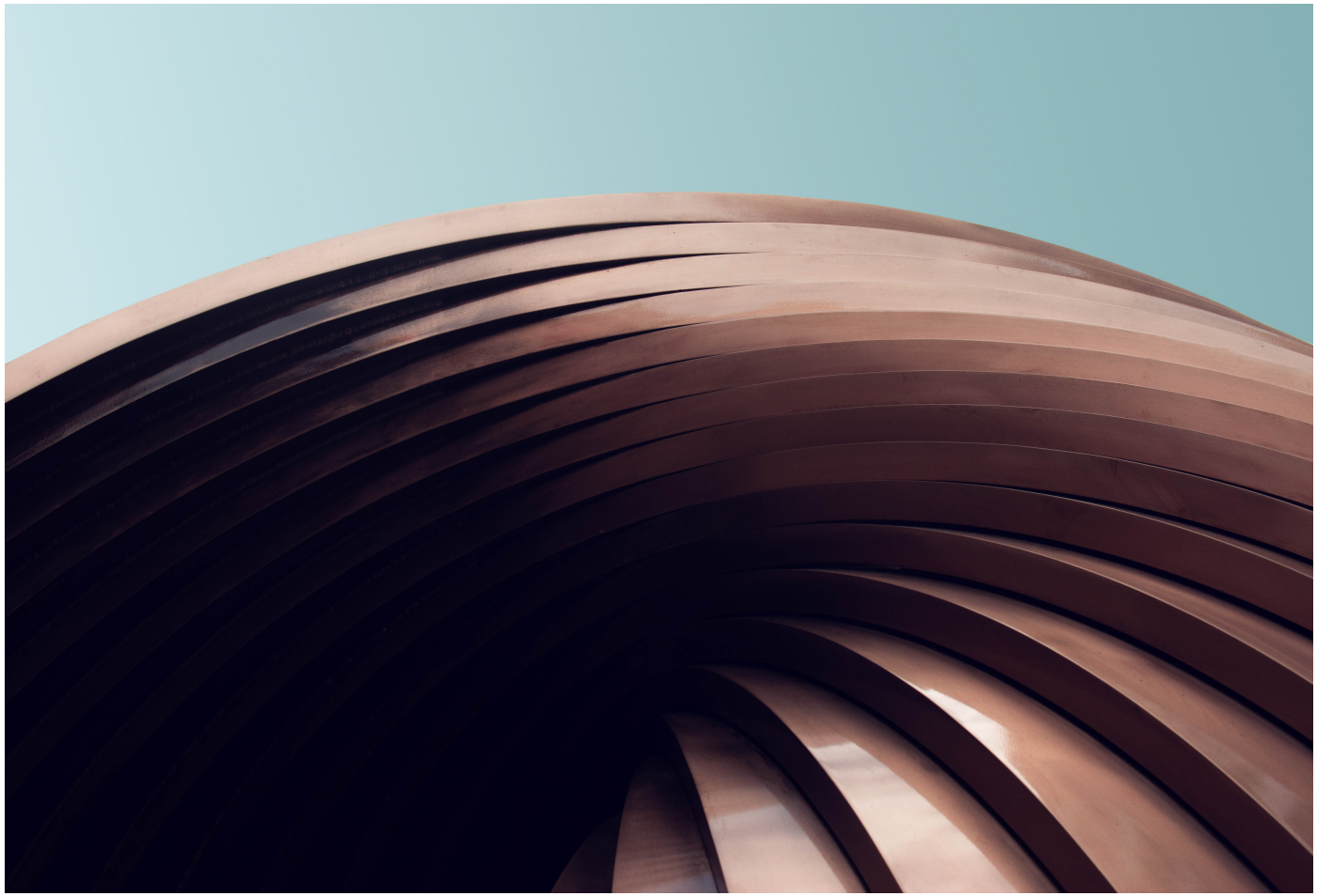
Intuitively, **probability mass is lost** in infinite length parse trees that never produce a finite string.

Even if probability of an infinite length tree tends to zero, there are uncountably infinitely many such trees.

This pathological case typically doesn't arise when PCFG rules are empirically estimated.

See next lecture for a presentation of such methods.

Lexicalised CFG



Nkululeko Jonas from Unsplash

Lexicalised CFG

In the rule VP $\rightarrow$ V NP, how do we know whether NP is a valid argument for a given V?

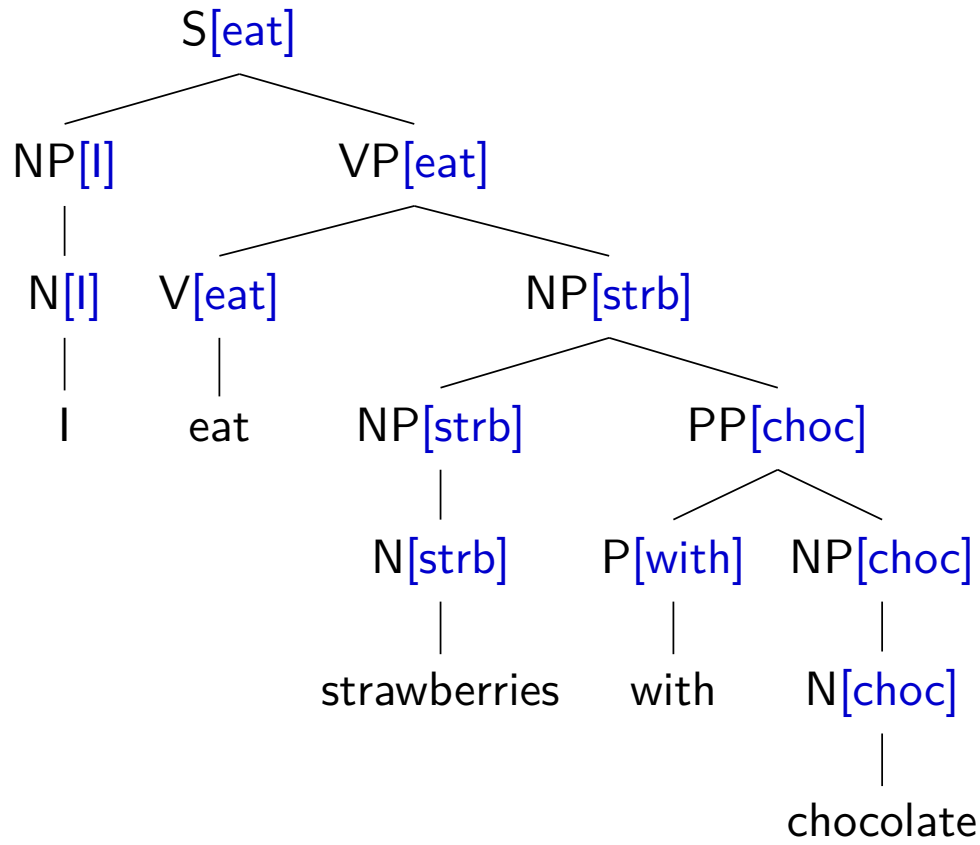
In our previous CFGs, each phrase records the head type but not the lexical content of the head. This makes the model **insensitive** to **lexical selection**, resulting in a loss in accuracy.

A CFG can model lexical relations by copying up heads through the whole structure of the phrase.

The result is called a **lexicalised context-free grammar** (LCFG).

Lexicalised CFG

Example : Parse tree from a LCFG



Lexicalised CFG

Example : In order to solve PP attachment, in a LCFG we may assign higher probability to rule

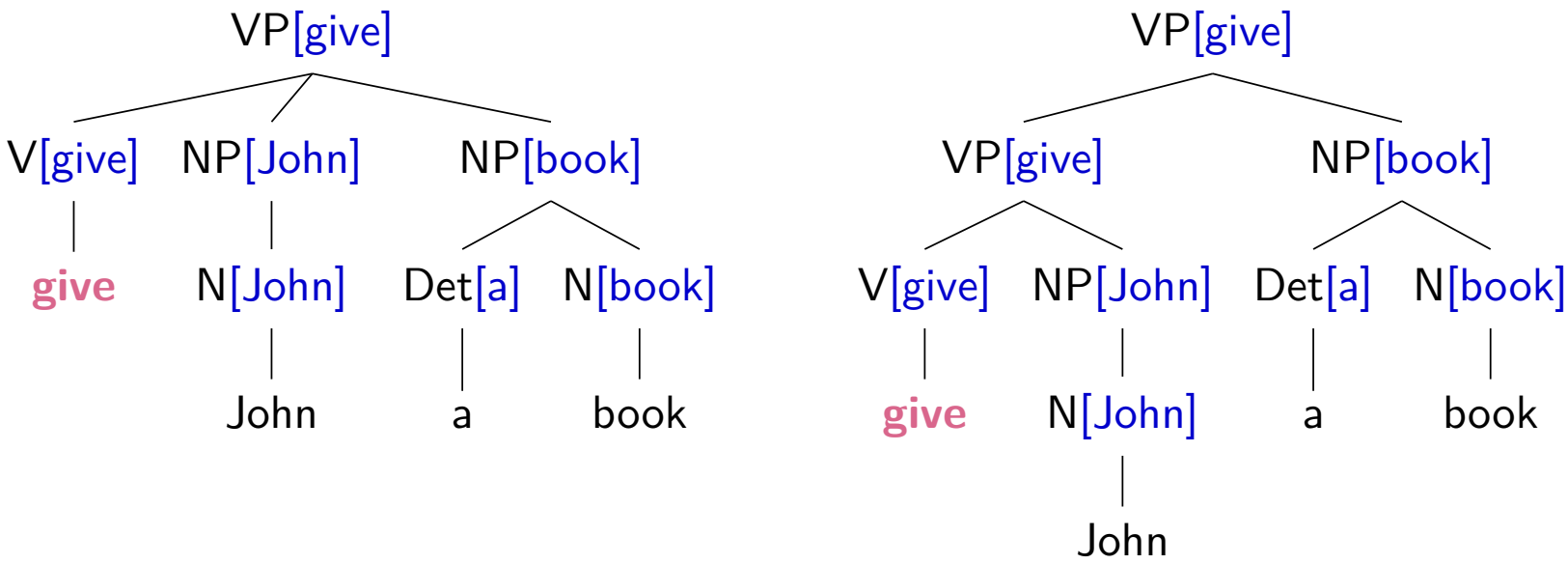
$$\text{NP[strb]} \quad \rightarrow \quad \text{NP[strb] PP[choc]}$$

than to rule

$$\text{NP[strb]} \quad \rightarrow \quad \text{NP[strb] PP[fork]}$$

Lexicalised CFG

It is very convenient to **binarize** a LCFG



Each rule has at most two lexical items (words). The result is called a **bilexical** CFG.

We are making an independence assumption for arguments and modifiers.

Compare also with 2-gram models.

Research papers

Title: Evidence against the context-freeness of natural language
Authors: Stuart Shieber
Journal: Linguistics and Philosophy 8 (3): 333–343 (1985)
Content: This paper offers evidence that natural language goes beyond the generative power of context-free grammars, based on data collected from Swiss-German.

`http://www.eecs.harvard.edu/shieber/Biblio/Papers/shieber85.pdf`