

# Natural Language Processing

AIMA Ch 23

# Additional Reference

---

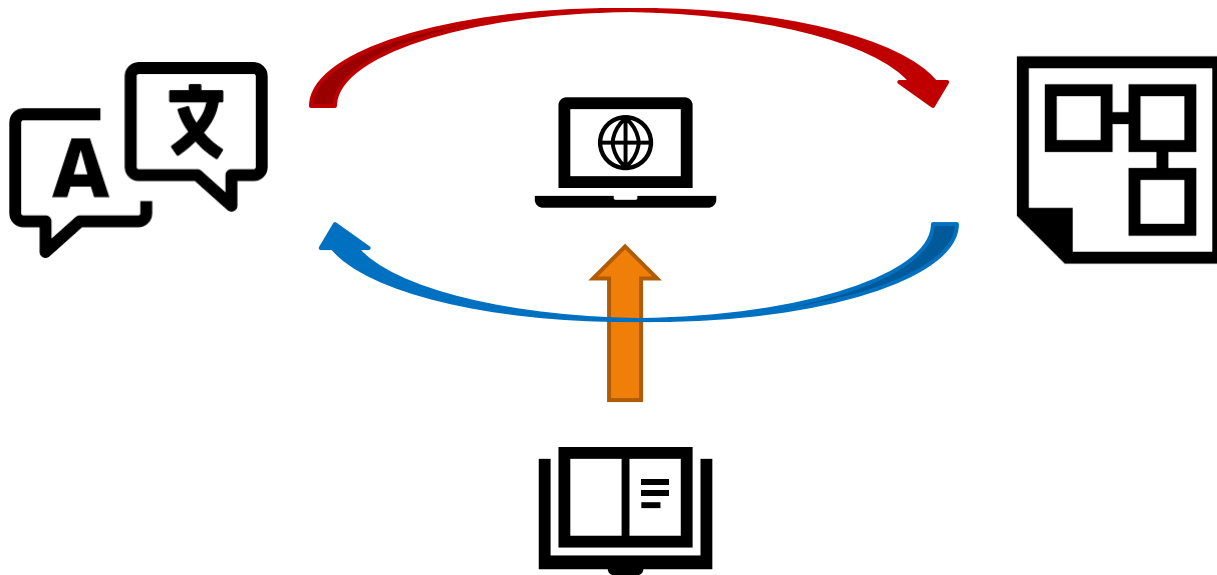
- ▶ [SLP] Speech and Language Processing, Daniel Jurafsky and James H. Martin
  - ▶ 2nd edition, 2008
  - ▶ 3rd edition, Sept. 2021
- ▶ Parsing
  - ▶ [AIMA] Ch 23
  - ▶ [SLP] Ch 12, 13, 14



# What is NLP?

---

- ▶ Automating the **analysis**, **generation**, and **acquisition** of human (“natural”) language



# NLP Applications

---

- ▶ ChatBot
  - ▶ Question answering, conversation
- ▶ Machine translation
- ▶ Information extraction
  - ▶ From financial and law documents, e-commerce websites, etc.
- ▶ Chinese IME
- ▶ Grammatical checker
- ▶ Essay scoring
- ▶ News generation



# Levels of NLP Research

---

|                         |   |
|-------------------------|---|
| Phonetics and phonology | knowledge about linguistic sounds   |
| Morphology              | knowledge of the meaningful components of words                                     |
| Syntax                  | knowledge of the structural relationships between words                             |
| Lexical semantics       | knowledge of word meaning   |
| Compositional semantics | knowledge of the meaning of sentences   |
| Pragmatics              | knowledge of the relationship of meaning to the goals and intentions of the speaker |
| Discourse               | knowledge about linguistic units larger than a single sentence                      |





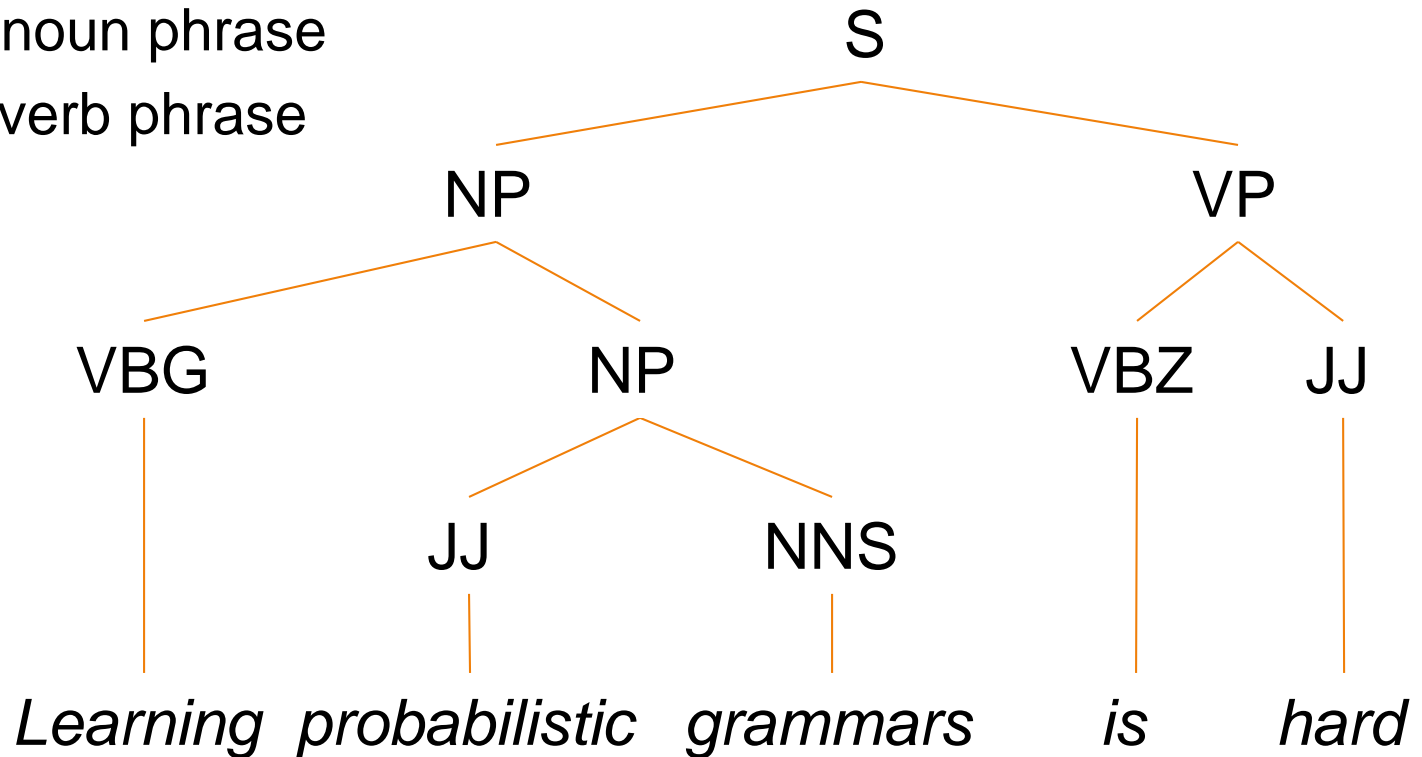
# Constituency Parsing



# Constituent parse tree

---

- ▶ Also called a phrase structure parse
- ▶ Each non-leaf node represents a phrase (i.e., constituent)
  - ▶ S: sentence
  - ▶ NP: noun phrase
  - ▶ VP: verb phrase
  - ▶ ...



# Grammars

---

- ▶ Grammar
  - ▶ the set of constituents and the rules that govern how they combine
- ▶ Lots of different theories of grammar
- ▶ Context-free grammars (CFGs)
  - ▶ Also known as: Phrase structure grammars
  - ▶ One of the simplest and most basic grammar formalisms





# Context-Free Grammars

---

- ▶ A context-free grammar has four components
  - ▶ A set  $\Sigma$  of terminals (words)
  - ▶ A set  $N$  of nonterminals (phrases)
  - ▶ A start symbol  $S \in N$
  - ▶ A set  $R$  of production rules
    - ▶ Specifies how a nonterminal can produce a string of terminals and/or nonterminals



# Example Grammar

| Grammar Rules   | Examples  |
|---|---|
| $S \rightarrow NP VP$   | I + want a morning flight   |
| $NP \rightarrow$<br>  <i>Pronoun</i><br>  <i>Proper-Noun</i><br>  <i>Det Nominal</i><br>$Nominal \rightarrow$<br>  <i>Nominal Noun</i><br>  <i>Noun</i> | I<br>Los Angeles<br>a + flight<br>morning + flight<br>flights                     |
| $VP \rightarrow$<br>  <i>Verb</i><br>  <i>Verb NP</i><br>  <i>Verb NP PP</i><br>  <i>Verb PP</i>  | do<br>want + a flight<br>leave + Boston + in the morning<br>leaving + on Thursday |
| $PP \rightarrow$ <i>Preposition NP</i>  | from + Los Angeles  |



# Example Grammar

---

*Noun* → *flights* | *breeze* | *trip* | *morning*  
*Verb* → *is* | *prefer* | *like* | *need* | *want* | *fly*  
*Adjective* → *cheapest* | *non-stop* | *first* | *latest*  
                  | *other* | *direct*  
*Pronoun* → *me* | *I* | *you* | *it*  
*Proper-Noun* → *Alaska* | *Baltimore* | *Los Angeles*  
                  | *Chicago* | *United* | *American*  
*Determiner* → *the* | *a* | *an* | *this* | *these* | *that*  
*Preposition* → *from* | *to* | *on* | *near*  
*Conjunction* → *and* | *or* | *but*



# Sentence Generation

---

- ▶ A grammar can be used to generate a string
  - ▶ starting from a string containing only the start symbol  $S$
  - ▶ recursively applying the rules to rewrite the string
  - ▶ until the string contains only terminals
- ▶ The generative process specifies the **grammatical structure (parse tree)** of the string



# Example

$S \rightarrow NP VP$

$S \rightarrow Aux NP VP$

$S \rightarrow VP$

$NP \rightarrow Pronoun$

$NP \rightarrow Proper-Noun$

$NP \rightarrow Det Nominal$

$NP \rightarrow Nominal$

$Nominal \rightarrow Noun$

$Nominal \rightarrow Nominal Noun$

$Nominal \rightarrow Nominal PP$

$VP \rightarrow Verb$

$VP \rightarrow Verb NP$

$VP \rightarrow Verb NP PP$

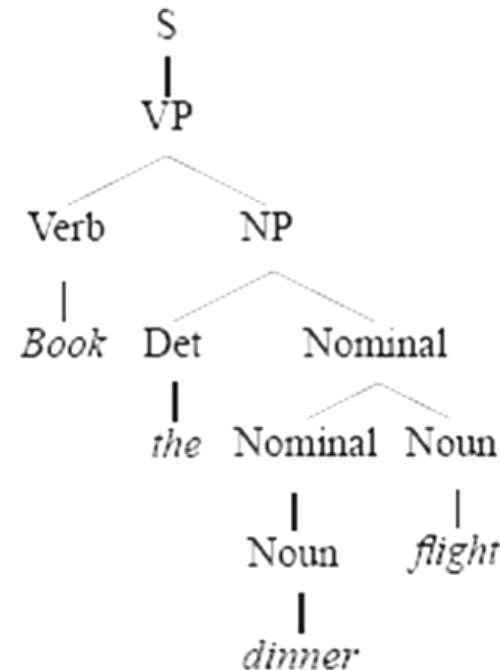
$VP \rightarrow Verb PP$

$VP \rightarrow Verb NP NP$

$VP \rightarrow VP PP$

$PP \rightarrow Preposition NP$

.....



*Book the dinner flight*



# Sentence Parsing

---

- ▶ Parsing is the process of taking a string and a grammar and returning one or more parse tree(s) for that string
  - ▶ If no parse tree can be found, then the string does not belong to the language
  - ▶ Parsing algorithms: CYK, Earley, etc.
    - ▶ To be introduced later



# Probabilistic Grammars

---

- ▶ Also called stochastic grammars
- ▶ Each rule is associated with a probability

$$\alpha \rightarrow \beta : P(\alpha \rightarrow \beta | \alpha)$$

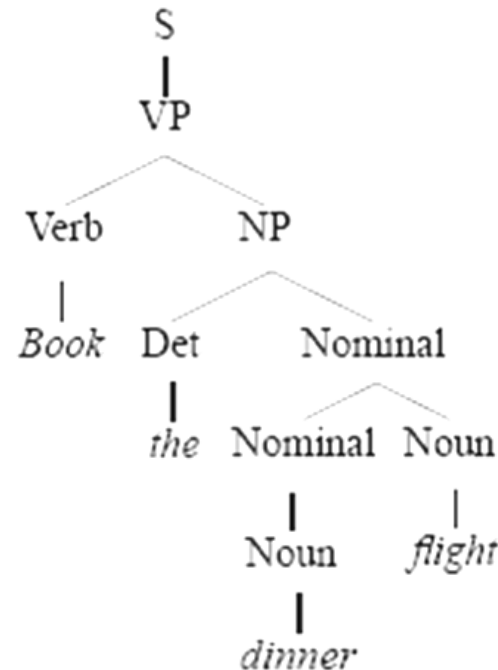
- ▶ The probability of a parse tree is the product of the probabilities of all the rules used in generating the parse tree



# Example

|                                    |       |
|------------------------------------|-------|
| $S \rightarrow NP VP$              | [.80] |
| $S \rightarrow Aux NP VP$          | [.15] |
| $S \rightarrow VP$                 | [.05] |
| $NP \rightarrow Pronoun$           | [.35] |
| $NP \rightarrow Proper-Noun$       | [.30] |
| $NP \rightarrow Det Nominal$       | [.20] |
| $NP \rightarrow Nominal$           | [.15] |
| $Nominal \rightarrow Noun$         | [.75] |
| $Nominal \rightarrow Nominal Noun$ | [.20] |
| $Nominal \rightarrow Nominal PP$   | [.05] |
| $VP \rightarrow Verb$              | [.35] |
| $VP \rightarrow Verb NP$           | [.20] |
| $VP \rightarrow Verb NP PP$        | [.10] |
| $VP \rightarrow Verb PP$           | [.15] |
| $VP \rightarrow Verb NP NP$        | [.05] |
| $VP \rightarrow VP PP$             | [.15] |
| $PP \rightarrow Preposition NP$    | [1.0] |

.....



*Book the dinner flight*

$$P(T) = .05 \times .20 \times .20 \times .20 \times .75 \times .30 \times .60 \times .10 \times .40 = 2.2 \times 10^{-6}$$





# Ambiguity

---

- ▶ A sentence is ambiguous if it has more than one possible parse tree
  - ▶ ...and hence more than one interpretation
- ▶ Examples
  - ▶ Astronomers saw stars with ears.



# Example

---

$S \rightarrow NP VP$  1.0

$PP \rightarrow P NP$  1.0

$VP \rightarrow V NP$  0.7

$VP \rightarrow VP PP$  0.3

$P \rightarrow \textit{with}$  1.0

$V \rightarrow \textit{saw}$  1.0

$NP \rightarrow NP PP$  0.4

$NP \rightarrow \textit{astronomers}$  0.1

$NP \rightarrow \textit{ears}$  0.18

$NP \rightarrow \textit{saw}$  0.04

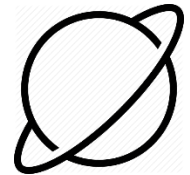
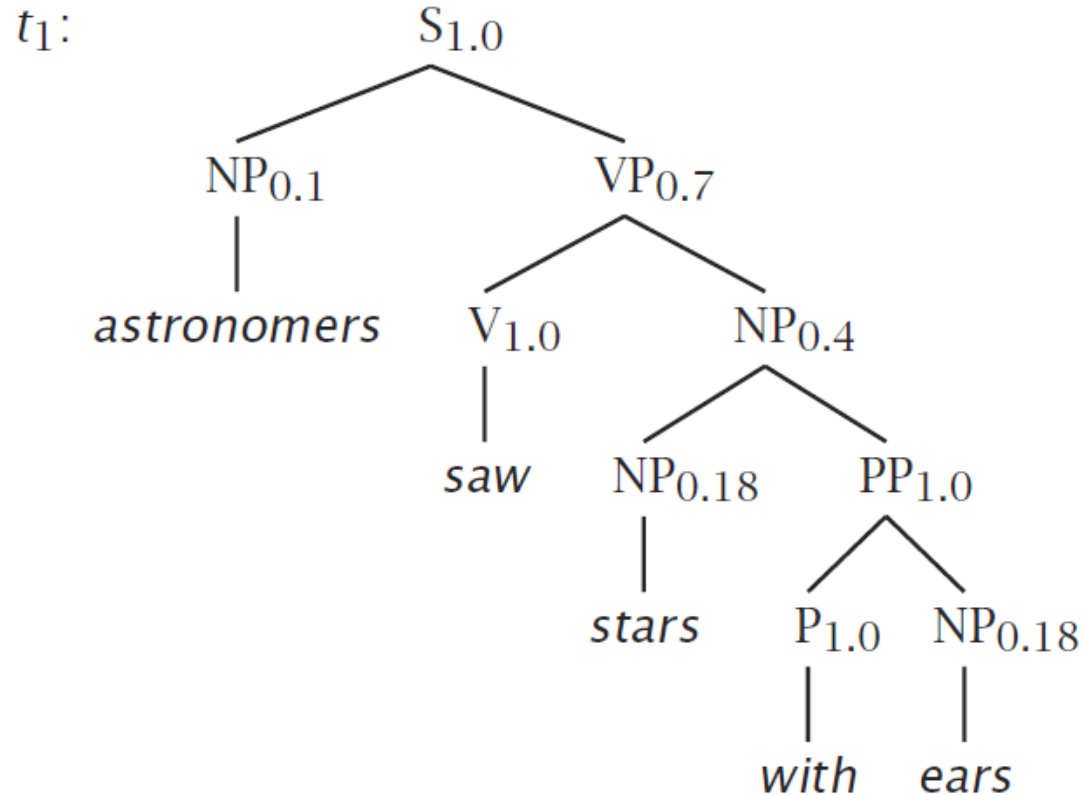
$NP \rightarrow \textit{stars}$  0.18

$NP \rightarrow \textit{telescopes}$  0.1



# Example

---

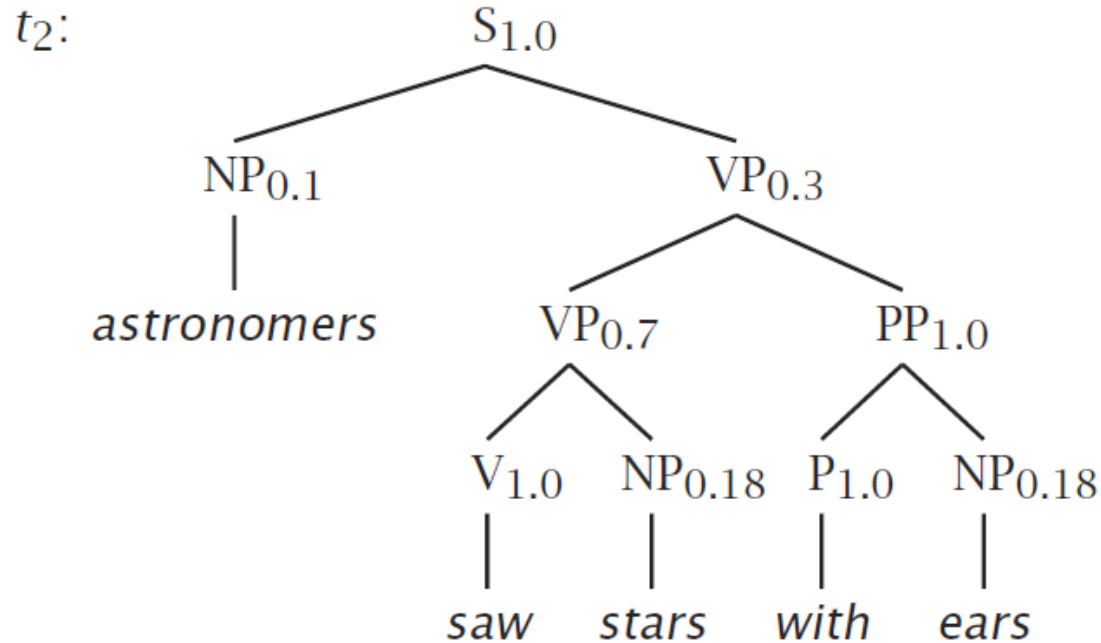


$$\begin{aligned} P(t_1) &= 1.0 \times 0.1 \times 0.7 \times 1.0 \times 0.4 \\ &\quad \times 0.18 \times 1.0 \times 1.0 \times 0.18 \\ &= 0.0009072 \end{aligned}$$



# Example

---



$$\begin{aligned} P(t_2) &= 1.0 \times 0.1 \times 0.3 \times 0.7 \times 1.0 \\ &\quad \times 0.18 \times 1.0 \times 1.0 \times 0.18 \\ &= 0.0006804 \end{aligned}$$





# Parsing Algorithm

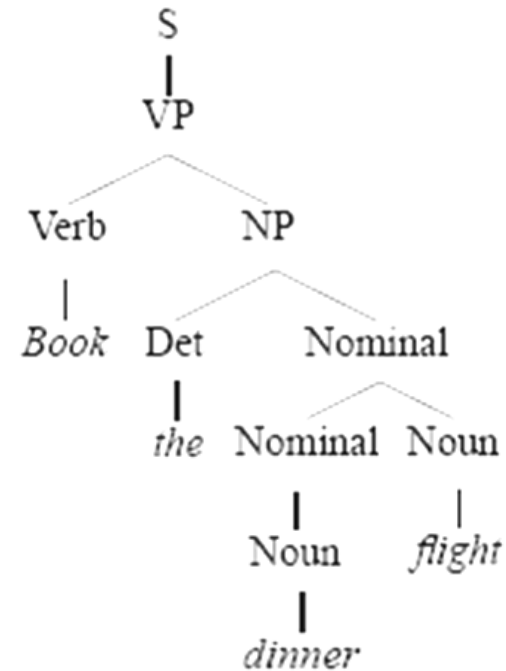


# Parsing

---

- ▶ Parsing with CFGs is the task of assigning proper parse trees to input strings

*Book the dinner flight*



# Parsing

---

- ▶ A brute-force approach
  - ▶ Enumerate all parse trees consistent with the input string
- ▶ Problem
  - ▶ Number of binary trees with  $n$  leaves is the Catalan number  $C_{n-1}$
  - ▶ (Exponential growth)



# Cocke–Younger–Kasami Algorithm (CYK)

---

- ▶ A bottom-up dynamic programming algorithm
- ▶ Applies to CFG in **Chomsky Normal Form (CNF)**
  - ▶ Only two types of production rules

$A \rightarrow B C$

$A \rightarrow w$

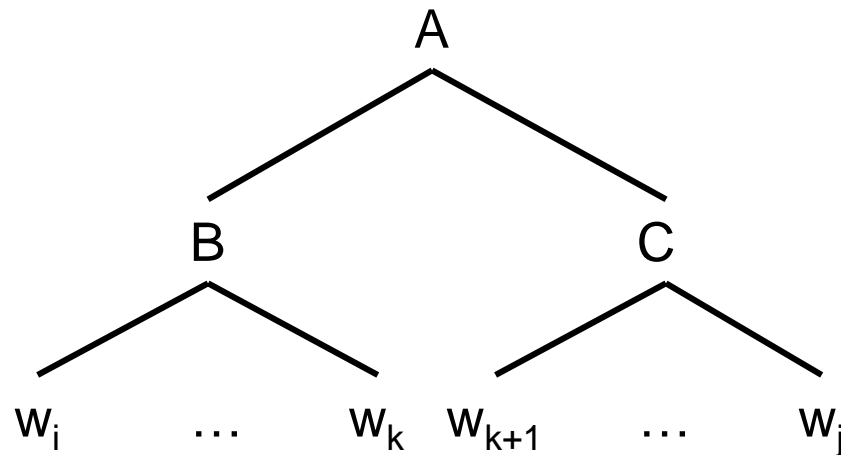




# Parsing

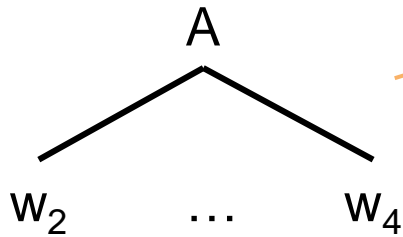
---

- ▶ Dynamic programming
  - ▶ Divide the problem into many sub-problems
    - ▶ Sub-problem: parsing the substring between positions  $i$  and  $j$
  - ▶ Solutions to smaller sub-problems are reused in solving larger sub-problems



# CYK

- ▶ Build a table so that a non-terminal  $A$  spanning from  $i$  to  $j$  in the input is placed in cell  $[i-1, j]$  in the table.



|   | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 0 |   |   |   |   |   |
| 1 |   |   |   | A |   |
| 2 |   |   |   |   |   |
| 3 |   |   |   |   |   |
| 4 |   |   |   |   |   |

- ▶ So a non-terminal spanning an entire string will sit in cell  $[0, n]$ 
  - ▶ Hopefully an  $S$

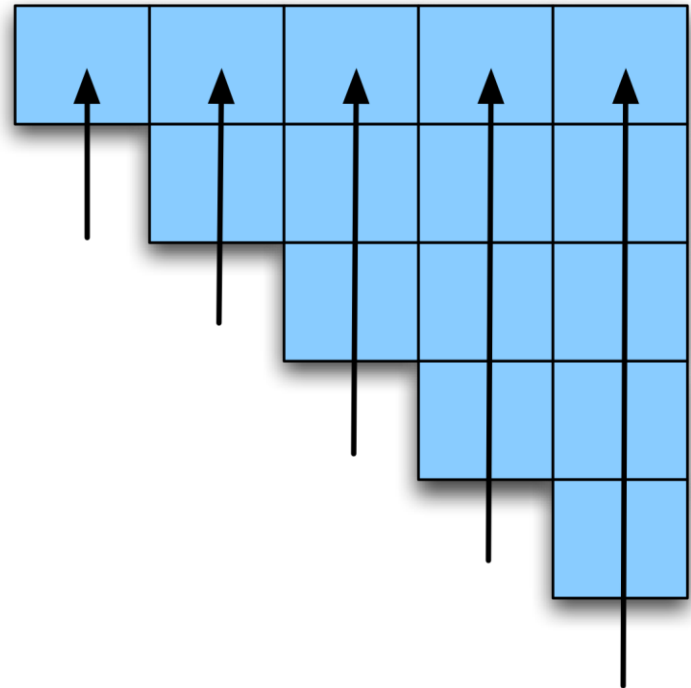


# Example

A completed table for input  
“Book the flight through  
Houston”

| <i>Book</i>                              | <i>the</i>   | <i>flight</i>             | <i>through</i> | <i>Houston</i>                  |
|--|--------------|---------------------------|----------------|---------------------------------|
| S, VP, Verb<br>Nominal,<br>Noun<br>[0,1] | [0,2]        | S,VP,X2<br>[0,3]          | [0,4]          | S,VP,X2<br>[0,5]                |
|  | Det<br>[1,2] | NP<br>[1,3]               | [1,4]          | NP<br>[1,5]                     |
|  |              | Nominal,<br>Noun<br>[2,3] | [2,4]          | Nominal<br>[2,5]                |
|  |              |                           | Prep<br>[3,4]  | PP<br>[3,5]                     |
|  |              |                           |                | NP,<br>Proper-<br>Noun<br>[4,5] |

We fill the table from  
bottom up



# CYK

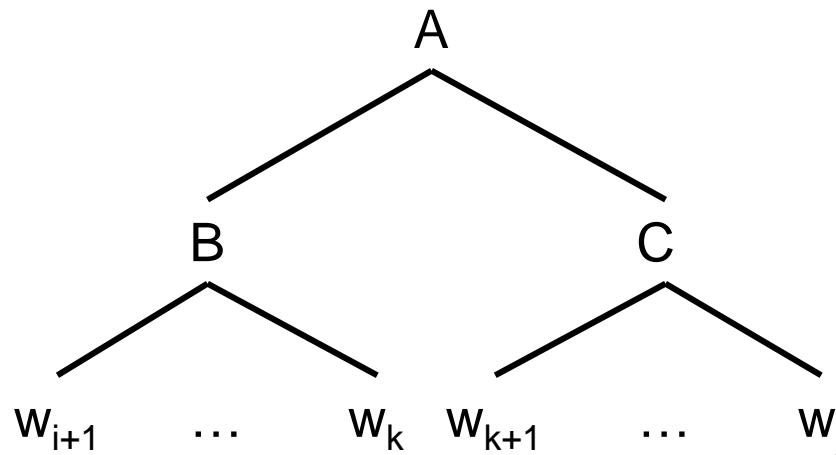
---

- ▶ Base case:

- ▶  $A$  is in cell  $[i-1, i]$  iff. there exists a rule  $A \rightarrow w_i$

- ▶ Recursion:

- ▶  $A$  is in cell  $[i, j]$  iff. for some rule  $A \rightarrow B C$  there is a  $B$  in cell  $[i, k]$  and a  $C$  in cell  $[k, j]$  for some  $k$ .



# CYK Algorithm

---

```
function CKY-PARSE(words, grammar) returns table  
  
  for  $j \leftarrow$  from 1 to LENGTH(words) do  
     $table[j-1, j] \leftarrow \{A \mid A \rightarrow words[j] \in grammar\}$   
    for  $i \leftarrow$  from  $j-2$  downto 0 do  
      for  $k \leftarrow i+1$  to  $j-1$  do  
         $table[i, j] \leftarrow table[i, j] \cup$   
           $\{A \mid A \rightarrow BC \in grammar,$   
             $B \in table[i, k],$   
             $C \in table[k, j]\}$ 
```



# CYK

---

- *The flight includes a meal.*

Grammar

- $S \rightarrow NP VP$
- $NP \rightarrow Det N$
- $VP \rightarrow V NP$
- $V \rightarrow \text{includes}$
- $Det \rightarrow \text{the}$
- $Det \rightarrow \text{a}$
- $N \rightarrow \text{meal}$
- $N \rightarrow \text{flight}$

|   | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 0 |   |   |   |   |   |
| 1 |   |   |   |   |   |
| 2 |   |   |   |   |   |
| 3 |   |   |   |   |   |
| 4 |   |   |   |   |   |



# CYK

---

- *The flight includes a meal.*

Grammar

- $S \rightarrow NP VP$
- $NP \rightarrow Det N$
- $VP \rightarrow V NP$
- $V \rightarrow \text{includes}$
- $Det \rightarrow \text{the}$
- $Det \rightarrow \text{a}$
- $N \rightarrow \text{meal}$
- $N \rightarrow \text{flight}$

|   | 1   | 2 | 3 | 4 | 5 |
|---|-----|---|---|---|---|
| 0 | Det |   |   |   |   |
| 1 |     |   |   |   |   |
| 2 |     |   |   |   |   |
| 3 |     |   |   |   |   |
| 4 |     |   |   |   |   |



# CYK

---

- *The **flight** includes a meal.*

Grammar

- $S \rightarrow NP VP$
- $NP \rightarrow Det N$
- $VP \rightarrow V NP$
- $V \rightarrow \text{includes}$
- $Det \rightarrow \text{the}$
- $Det \rightarrow \text{a}$
- $N \rightarrow \text{meal}$
- $N \rightarrow \text{flight}$

|   | 1   | 2 | 3 | 4 | 5 |
|---|-----|---|---|---|---|
| 0 | Det |   |   |   |   |
| 1 |     | N |   |   |   |
| 2 |     |   |   |   |   |
| 3 |     |   |   |   |   |
| 4 |     |   |   |   |   |





# CYK

---

- *The flight includes a meal.*

Grammar

- $S \rightarrow NP VP$
- $NP \rightarrow Det N$
- $VP \rightarrow V NP$
- $V \rightarrow \text{includes}$
- $Det \rightarrow \text{the}$
- $Det \rightarrow \text{a}$
- $N \rightarrow \text{meal}$
- $N \rightarrow \text{flight}$

|   | 1   | 2  | 3 | 4 | 5 |
|---|-----|----|---|---|---|
| 0 | Det | NP |   |   |   |
| 1 |     | N  |   |   |   |
| 2 |     |    |   |   |   |
| 3 |     |    |   |   |   |
| 4 |     |    |   |   |   |



# CYK

---

- *The flight includes a meal.*

Grammar

- $S \rightarrow NP VP$
- $NP \rightarrow Det N$
- $VP \rightarrow V NP$
- $V \rightarrow \text{includes}$
- $Det \rightarrow \text{the}$
- $Det \rightarrow \text{a}$
- $N \rightarrow \text{meal}$
- $N \rightarrow \text{flight}$

|   | 1   | 2  | 3 | 4 | 5 |
|---|-----|----|---|---|---|
| 0 | Det | NP |   |   |   |
| 1 |     | N  |   |   |   |
| 2 |     |    | V |   |   |
| 3 |     |    |   |   |   |
| 4 |     |    |   |   |   |



# CYK

---

- *The flight includes a meal.*

Grammar

- $S \rightarrow NP VP$
- $NP \rightarrow Det N$
- $VP \rightarrow V NP$
- $V \rightarrow \text{includes}$
- $Det \rightarrow \text{the}$
- $Det \rightarrow \text{a}$
- $N \rightarrow \text{meal}$
- $N \rightarrow \text{flight}$

|   | 1   | 2  | 3 | 4   | 5 |
|---|-----|----|---|-----|---|
| 0 | Det | NP |   |     |   |
| 1 |     | N  |   |     |   |
| 2 |     |    | V |     |   |
| 3 |     |    |   | Det |   |
| 4 |     |    |   |     |   |



# CYK

---

- *The flight includes a meal.*

Grammar

- $S \rightarrow NP VP$
- $NP \rightarrow Det N$
- $VP \rightarrow V NP$
- $V \rightarrow \text{includes}$
- $Det \rightarrow \text{the}$
- $Det \rightarrow \text{a}$
- $N \rightarrow \text{meal}$
- $N \rightarrow \text{flight}$

|   | 1   | 2  | 3 | 4   | 5 |
|---|-----|----|---|-----|---|
| 0 | Det | NP |   |     |   |
| 1 |     | N  |   |     |   |
| 2 |     |    | V |     |   |
| 3 |     |    |   | Det |   |
| 4 |     |    |   |     | N |



# CYK

---

- *The flight includes a meal.*

Grammar

- $S \rightarrow NP VP$
- $NP \rightarrow Det N$
- $VP \rightarrow V NP$
- $V \rightarrow \text{includes}$
- $Det \rightarrow \text{the}$
- $Det \rightarrow \text{a}$
- $N \rightarrow \text{meal}$
- $N \rightarrow \text{flight}$

|   | 1   | 2  | 3 | 4   | 5  |
|---|-----|----|---|-----|----|
| 0 | Det | NP |   |     |    |
| 1 |     | N  |   |     |    |
| 2 |     |    | V |     |    |
| 3 |     |    |   | Det | NP |
| 4 |     |    |   |     | N  |



# CYK

---

- *The flight includes a meal.*

Grammar

- $S \rightarrow NP VP$
- $NP \rightarrow Det N$
- $VP \rightarrow V NP$
- $V \rightarrow \text{includes}$
- $Det \rightarrow \text{the}$
- $Det \rightarrow \text{a}$
- $N \rightarrow \text{meal}$
- $N \rightarrow \text{flight}$

|   | 1   | 2  | 3 | 4   | 5  |
|---|-----|----|---|-----|----|
| 0 | Det | NP |   |     |    |
| 1 |     | N  |   |     |    |
| 2 |     |    | V |     | VP |
| 3 |     |    |   | Det | NP |
| 4 |     |    |   |     | N  |



# CYK

---

- *The flight includes a meal.*

Grammar

- $S \rightarrow NP VP$
- $NP \rightarrow Det N$
- $VP \rightarrow V NP$
- $V \rightarrow \text{includes}$
- $Det \rightarrow \text{the}$
- $Det \rightarrow \text{a}$
- $N \rightarrow \text{meal}$
- $N \rightarrow \text{flight}$

|   | 1   | 2  | 3 | 4   | 5  |
|---|-----|----|---|-----|----|
| 0 | Det | NP |   |     | S  |
| 1 |     | N  |   |     |    |
| 2 |     |    | V |     | VP |
| 3 |     |    |   | Det | NP |
| 4 |     |    |   |     | N  |



# CYK Parsing

---

- ▶ Is that really a parser?
  - ▶ We want a parse tree, not a yes/no answer
- ▶ Simple changes
  - ▶ Add back-pointers so that each state knows where it came from.
  - ▶ After filling the table, recursively retrieve the constituents from the top (i.e., the start symbol) down





# CYK

- *The flight includes a meal.*

Grammar

- $S \rightarrow NP VP$
- $NP \rightarrow Det N$
- $VP \rightarrow V NP$
- $V \rightarrow \text{includes}$
- $Det \rightarrow \text{the}$
- $Det \rightarrow \text{a}$
- $N \rightarrow \text{meal}$
- $N \rightarrow \text{flight}$

|   | 1   | 2  | 3 | 4   | 5  |
|---|-----|----|---|-----|----|
| 0 | Det | NP |   |     | S  |
| 1 |     | N  |   |     |    |
| 2 |     |    | V |     | VP |
| 3 |     |    |   | Det | NP |
| 4 |     |    |   |     | N  |

# Probabilistic Parsing

---

- ▶ What if we have a PCFG, and we want to find the parse tree of an input string with the highest probability?
- ▶ Still run CYK, but:
  - ▶ In cell  $[i-1, j]$  of the table, associate each nonterminal  $A$  with the probability of the best parse tree rooted at  $A$  covering substring from  $i$  to  $j$
  - ▶ The probabilities can be computed with a bottom-up recursive formula during CYK steps





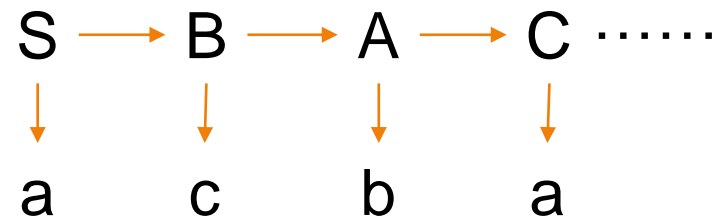
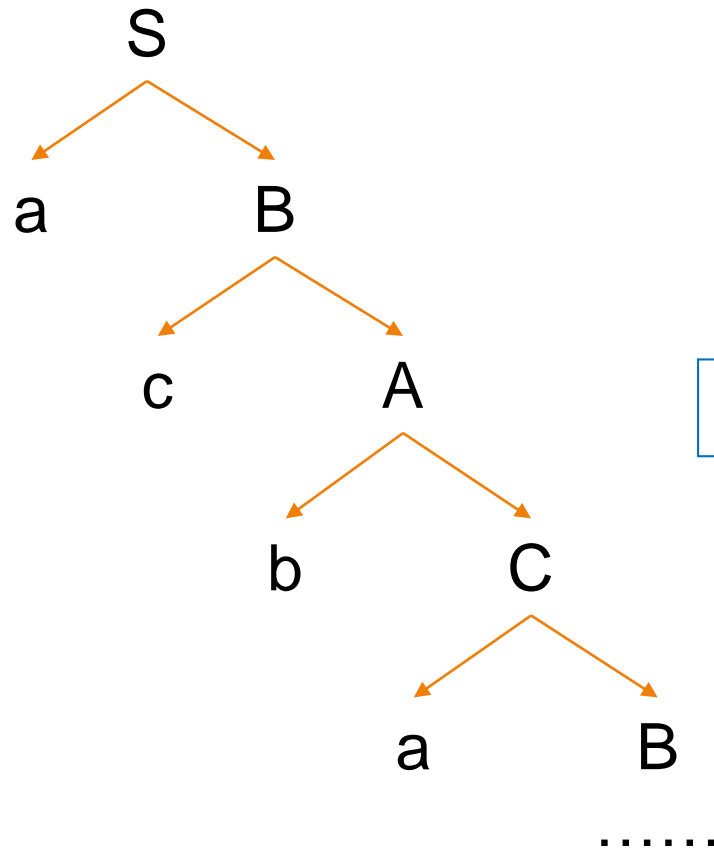
# Regular Grammar



# Regular Grammars

---

- ▶ Production rules are of the form  $A \rightarrow aB$  or  $A \rightarrow a$

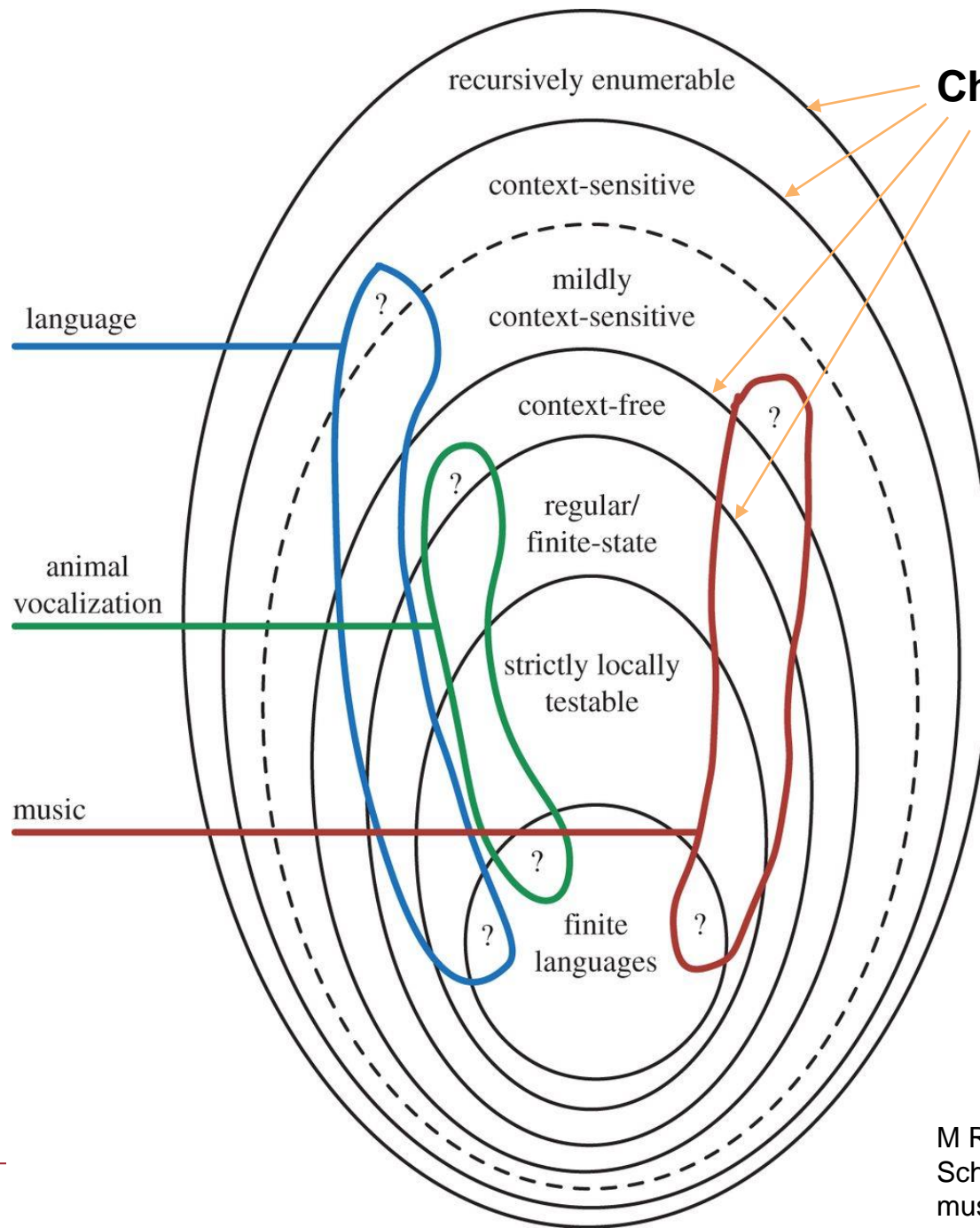
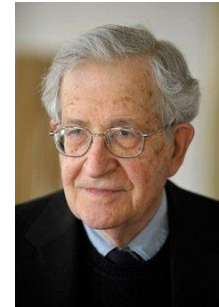


HMM = Probabilistic RG  $\subset$  PCFG

Viterbi  $\sim$  Probabilistic CYK



# Chomsky Hierarchy



M Rohrmeier, W Zuidema, G Wiggins, C Scharff. Principles of structure building in music, language and animal song.



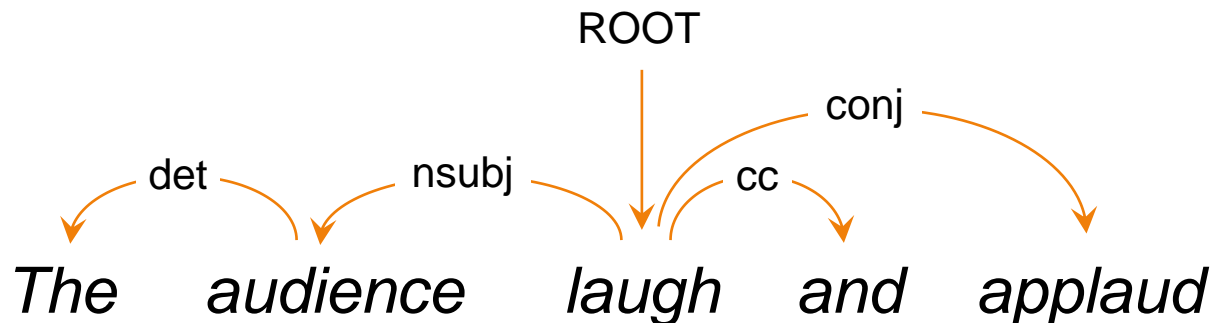
# Dependency Parsing



# Dependency Parse

---

- ▶ A dependency parse is a directed tree where
  - ▶ the nodes are the words in a sentence
    - ▶ ROOT: a special root node
  - ▶ The links between the words represent their dependency relations
    - ▶ Typically drawn as a directed arc from **head** to **dependent**
    - ▶ Dependency arcs may be typed (labeled)



# Dependency Types

---

| Argument Dependencies | Description            |
|-----------------------|------------------------|
| nsubj                 | nominal subject        |
| csbj                  | clausal subject        |
| dobj                  | direct object          |
| iobj                  | indirect object        |
| pobj                  | object of preposition  |
| Modifier Dependencies | Description            |
| tmod                  | temporal modifier      |
| appos                 | appositional modifier  |
| det                   | determiner             |
| prep                  | prepositional modifier |





# Dependency Parsing

---

## ▶ Advantages

- ▶ Deals well with free word order languages where the constituent structure is quite fluid
  - ▶ Ex: Czech, Turkish
- ▶ Dependency parses of sentences having the same meaning are more similar across languages than constituency parses
- ▶ Dependency structure often captures the syntactic relations needed by later applications
- ▶ Parsing can be faster than CFG-bases parsers



# Dependency Parsing

---

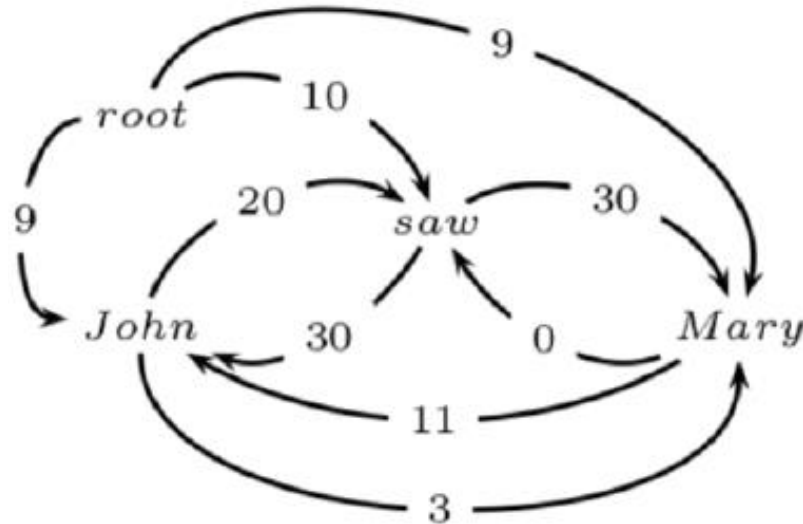
- ▶ Parsing
  - ▶ Taking a string and a grammar and returning one or more parse tree(s) for that string
- ▶ Probabilistic parsing
  - ▶ Find the highest-scoring parse tree
- ▶ Several approaches to dependency parsing
  - ▶ Next: a brief intro of graph-based parsing



# Graph-based parsing

---

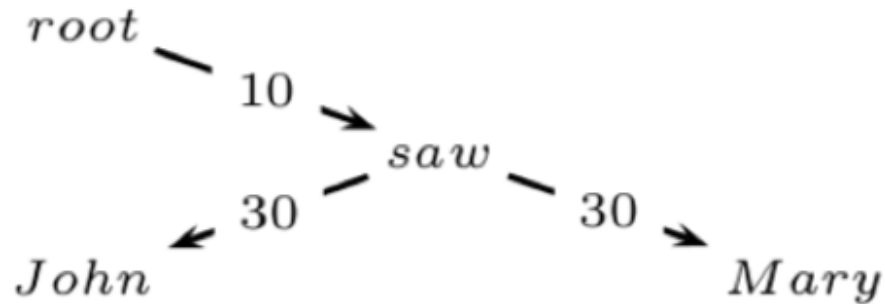
- ▶ Each arc has a non-negative score.
- ▶ An arc score is often computed from features of the two words and the context.



# Graph-based parsing

---

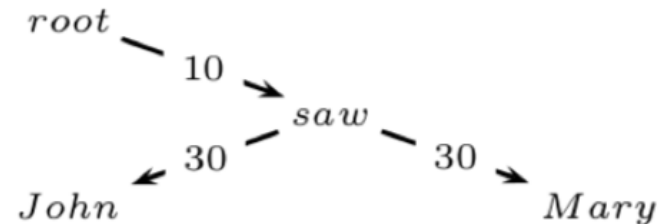
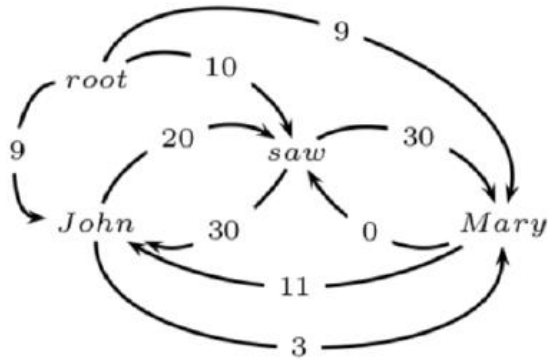
- ▶ The tree score is the product of arc scores.



# Graph-based parsing

---

- ▶ Parsing: find the highest-scoring parse tree
  - ▶ = max spanning directed tree (arborescence)
  - ▶ Solvable by greedy algorithm (Chu-Liu-Edmonds) and dynamic programming (Eisner)



# Dependency Grammar vs. CFG

# DG vs. CFG

---

- ▶ Dependency grammars are a subclass of CFGs

$a \rightarrow b$

$a \rightarrow c$

$d \leftarrow a$

$e \leftarrow a$



$A \rightarrow AB$

$A \rightarrow AC$

$A \rightarrow DA$

$A \rightarrow EA$

$A \rightarrow a$

$B \rightarrow b$

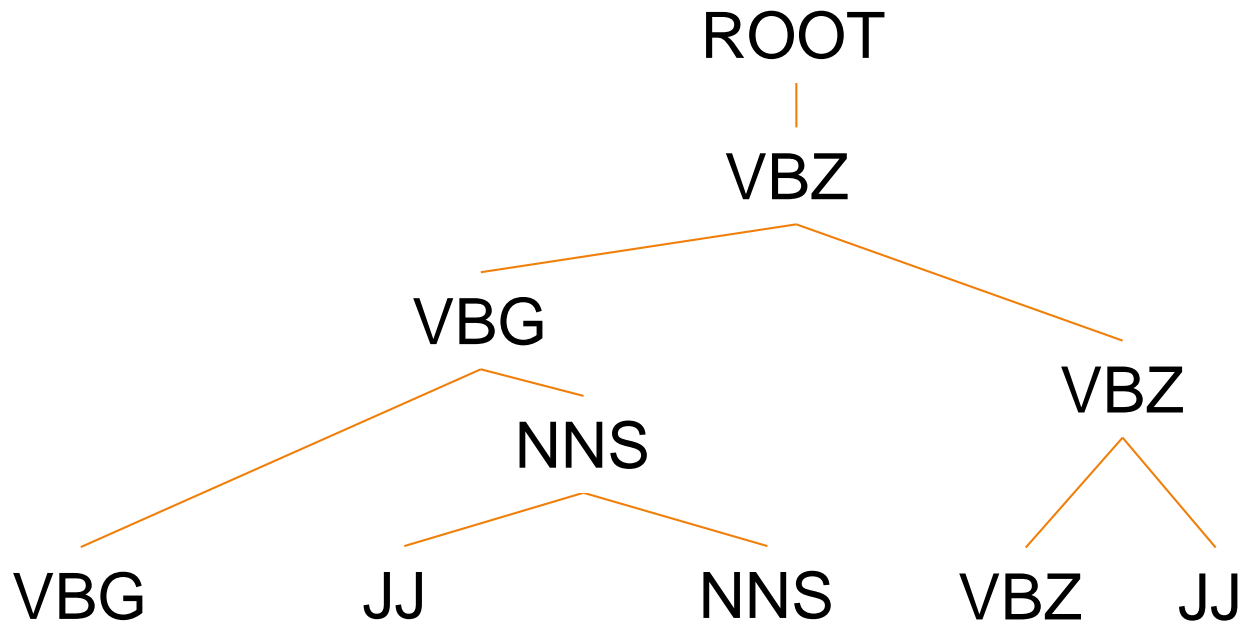
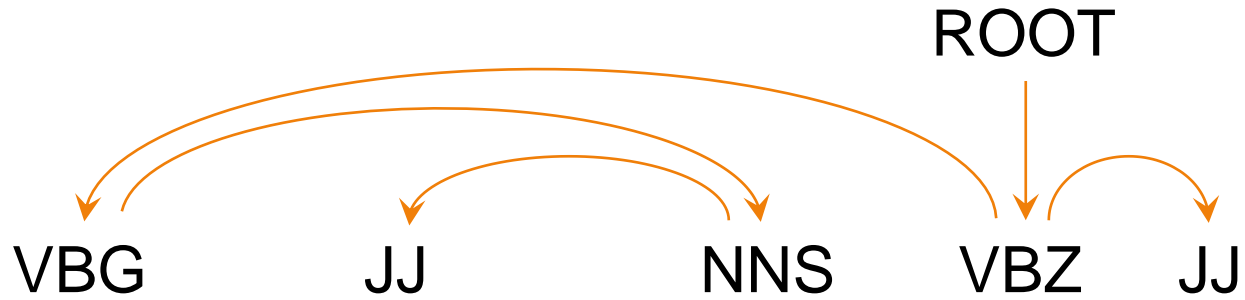
$C \rightarrow c$

$D \rightarrow d$

$E \rightarrow e$

# DG vs. CFG

---

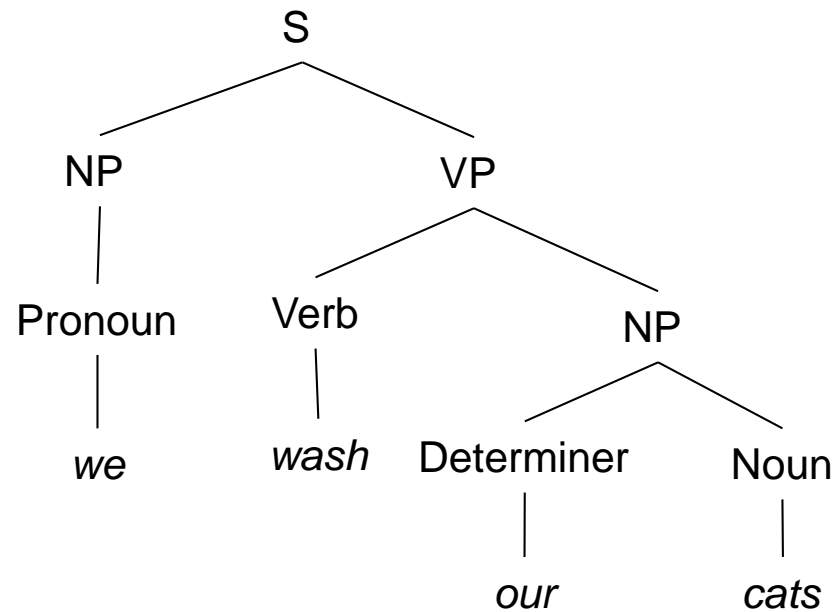




# Conversion

---

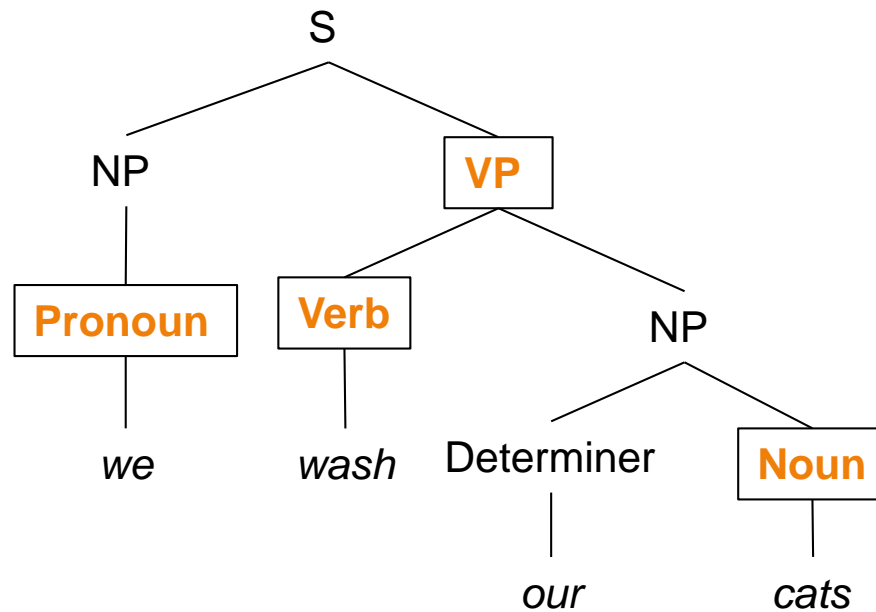
- ▶ From a constituent tree to a dependency tree
  - ▶ Constituent tree



# Conversion

---

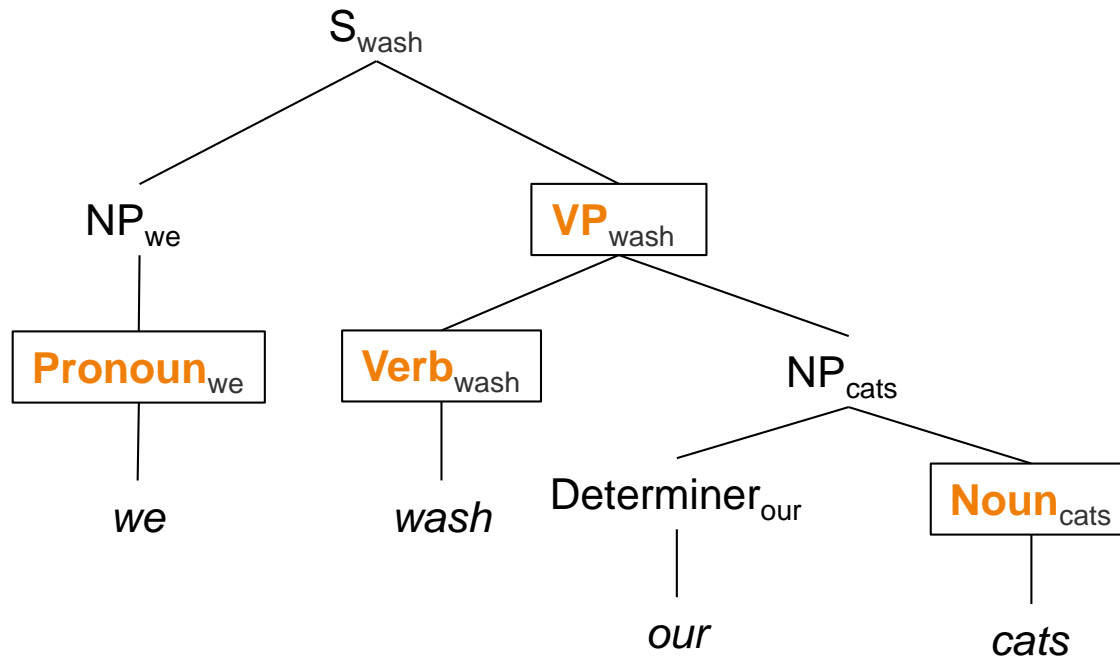
- ▶ From a constituent tree to a dependency tree
  - ▶ Constituent tree with heads



# Conversion

---

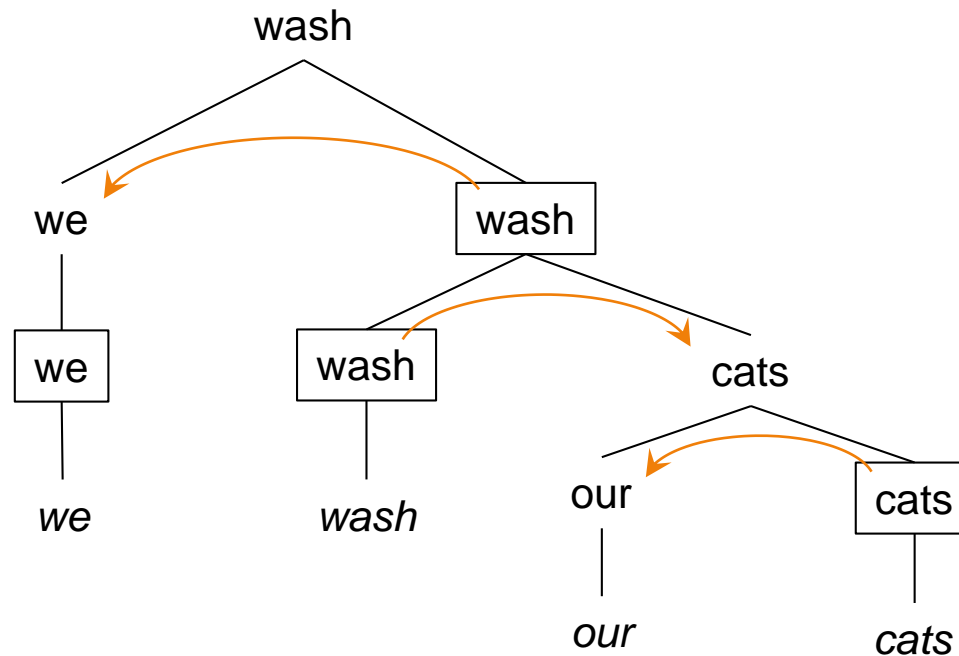
- ▶ From a constituent tree to a dependency tree
  - ▶ Constituent tree with heads, lexicalized



# Constituency to dependency

---

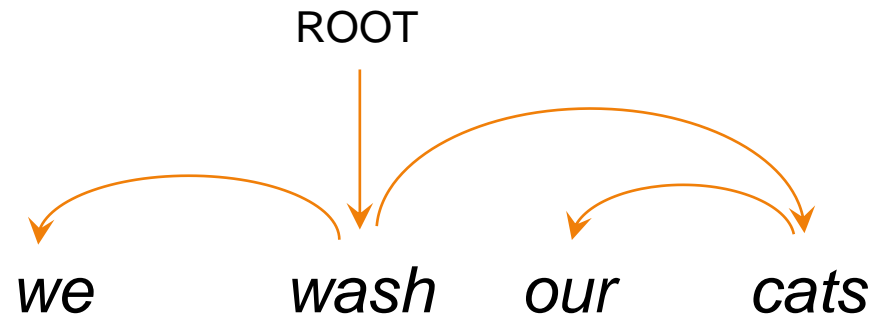
- ▶ From a constituent tree to a dependency tree
  - ▶ Constituent tree with heads, lexicalized



# Conversion

---

- ▶ From a constituent tree to a dependency tree
  - ▶ Dependency tree



# Summary

---

- ▶ Constituency parsing
  - ▶ (Probabilistic) context-free grammars
  - ▶ CYK algorithm
- ▶ Regular grammars
- ▶ Dependency parsing

