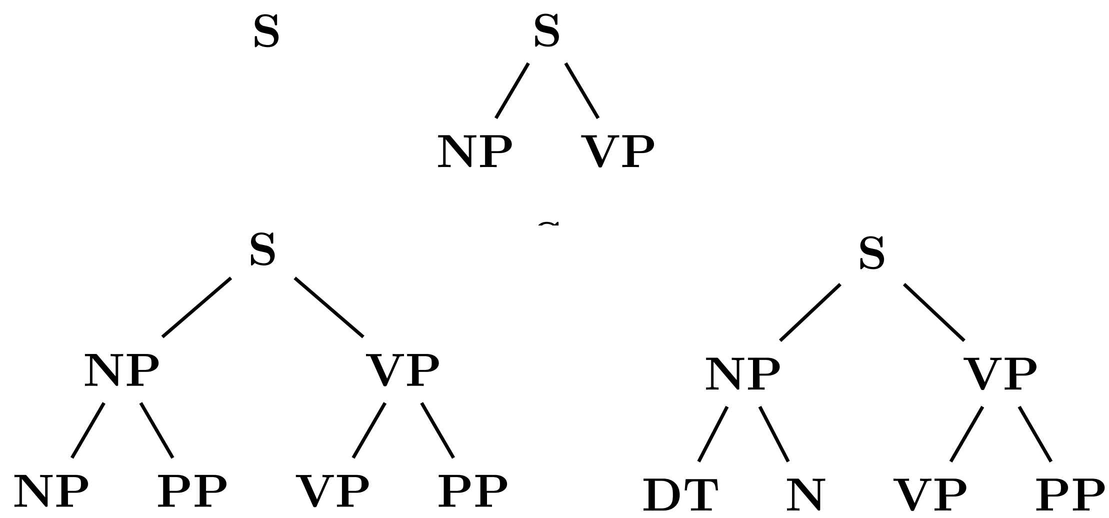
CS498JH: Introduction to NLP (Fall 2012)
http://cs.illinois.edu/class/cs498jh

Lecture 12: The CKY parsing algorithm

Julia Hockenmaier
juliahmr@illinois.edu
3324 Siebel Center
Office Hours: Wednesday, 12:15-1:15pm

Naive top-down parsing

S	→	**NP VP**
NP	→	**NP PP**
NP	→	**Noun**
VP	→	**VP PP**
VP	→	**Verb NP**

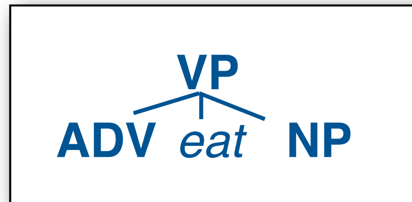


The number of trees is exponential!
Many subtrees are the same.

Chomsky Normal Form

The right-hand side of a standard CFG can have an **arbitrary number of symbols** (terminals and nonterminals):

VP → ADV eat NP



A CFG in **Chomsky Normal Form** (CNF) allows only two kinds of right-hand sides:

– **Two nonterminals:** VP → ADV VP

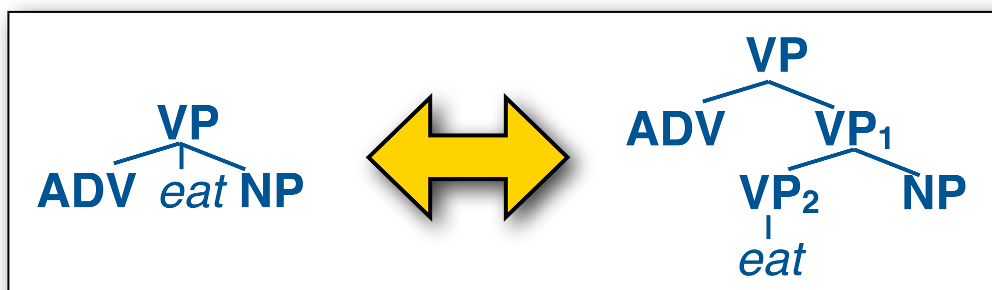
– **One terminal:** VP → eat

Any CFG can be transformed into an equivalent CNF:

$\mathrm{VP} \rightarrow \mathrm{ADVP}\ \mathbf{VP_1}$

$\mathbf{VP_1} \rightarrow \mathbf{VP_2}\ \mathrm{NP}$

$\mathbf{VP_2} \rightarrow \mathrm{eat}$



A note about ε-productions

Formally, context-free grammars are allowed to have **empty productions** (ε = the empty string):

VP → V NP   NP → DT Noun   NP → ε

These can always be **eliminated** without changing the language generated by the grammar:

VP → V NP   NP → DT Noun   NP → ε

becomes

VP → V NP   VP → V ε   NP → DT Noun

which in turn becomes

VP → V NP   VP → V   NP → DT Noun

We will assume that our grammars don't have ε-productions

CKY chart parsing algorithm

Bottom-up parsing:

start with the words

Dynamic programming:

save the results in a table/chart

re-use these results in finding larger constituents

Complexity: $O(n^3|G|)$

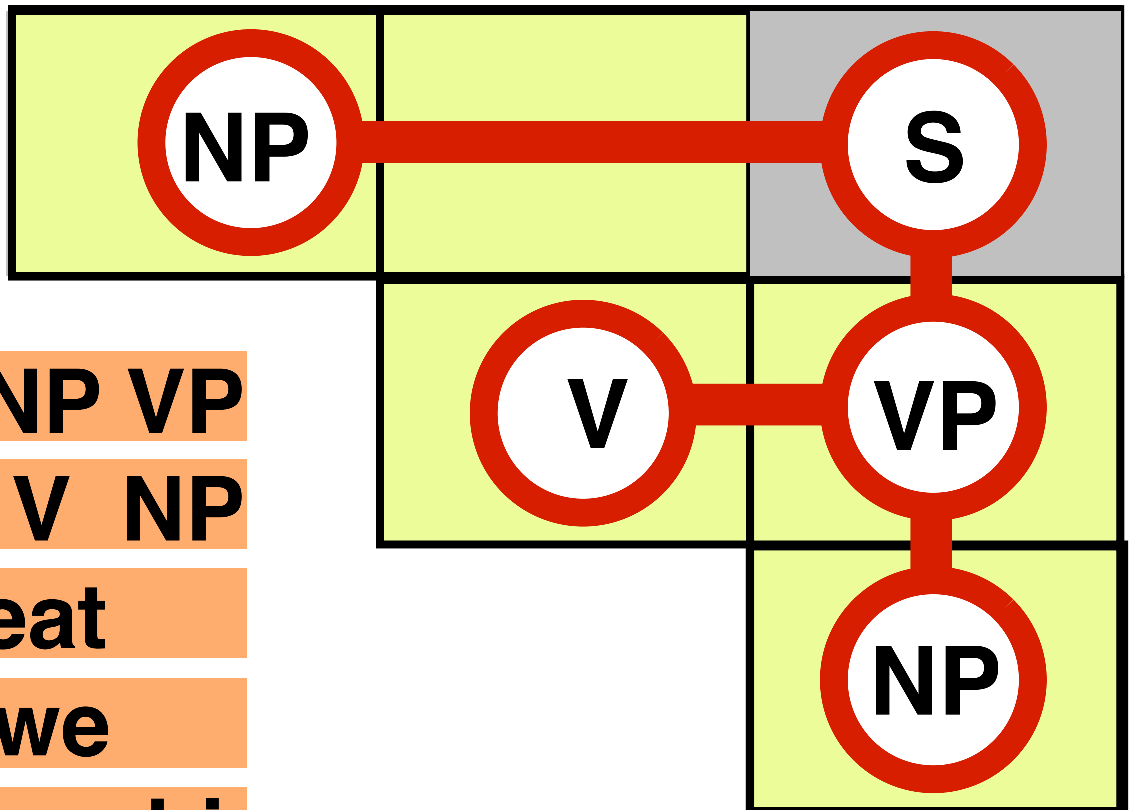
n: length of string, $|G|$: size of grammar)

Presumes a CFG in Chomsky Normal Form:

Rules are all either **A → B C** or **A → a**

(with **A,B,C** nonterminals and **a** a terminal)

The CKY parsing algorithm



To recover the parse tree, each entry needs **pairs** of backpointers.

S	**→**	**NP VP**
VP	**→**	**V NP**
V	**→**	**eat**
NP	**→**	**we**
NP	**→**	**sushi**

We eat sushi

The CKY parsing algorithm

we	we eat	we eat sushi	we eat sushi with	we eat sushi with tuna
	V eat	VP eat sushi	eat sushi with	VP eat sushi with tuna
			th	NP sushi with tuna
				PP with tuna
				tuna

S → NP VP
VP → V NP
VP → VP PP
V → eat
NP → NP PP
NP → we
NP → sushi
NP → tuna
PP → P NP
P → with

Each cell contains only a **single entry** for each nonterminal.
Each entry may have a **list** of pairs of backpointers.

We eat sushi with tuna

How do you count the **number of parse trees** for a sentence?

1. For each **pair of backpointers** (e.g. $VP \rightarrow V\ NP$): **multiply** #trees of children

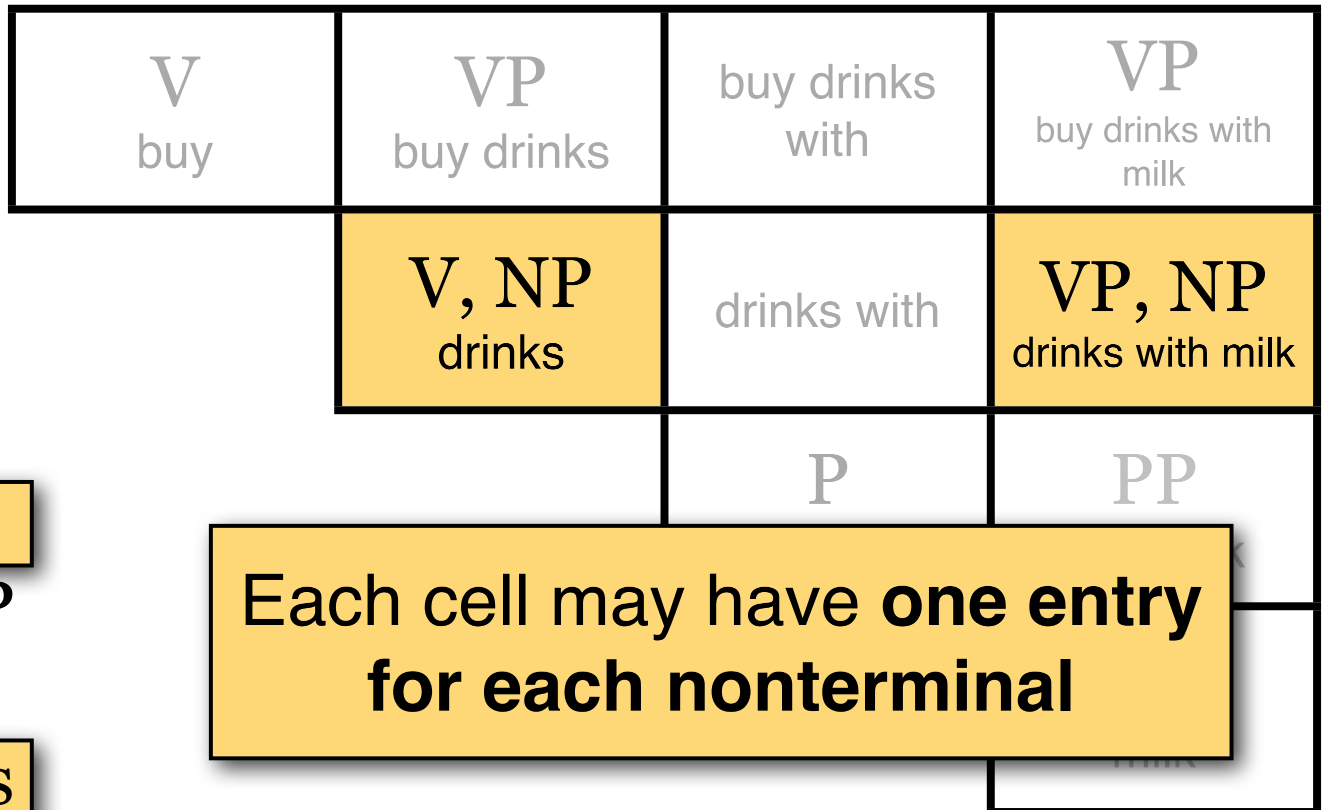
$$\text{trees}(VP_{VP \rightarrow V\ NP}) = \text{trees}(V) \times \text{trees}(NP)$$

2. For each **list of pairs of backpointers** (e.g. $VP \rightarrow V\ NP$ and $VP \rightarrow VP\ PP$): **sum** #trees

$$\text{trees}(VP) = \text{trees}(VP_{VP \rightarrow V\ NP}) + \text{trees}(VP_{VP \rightarrow VP\ PP})$$

The CKY parsing algorithm

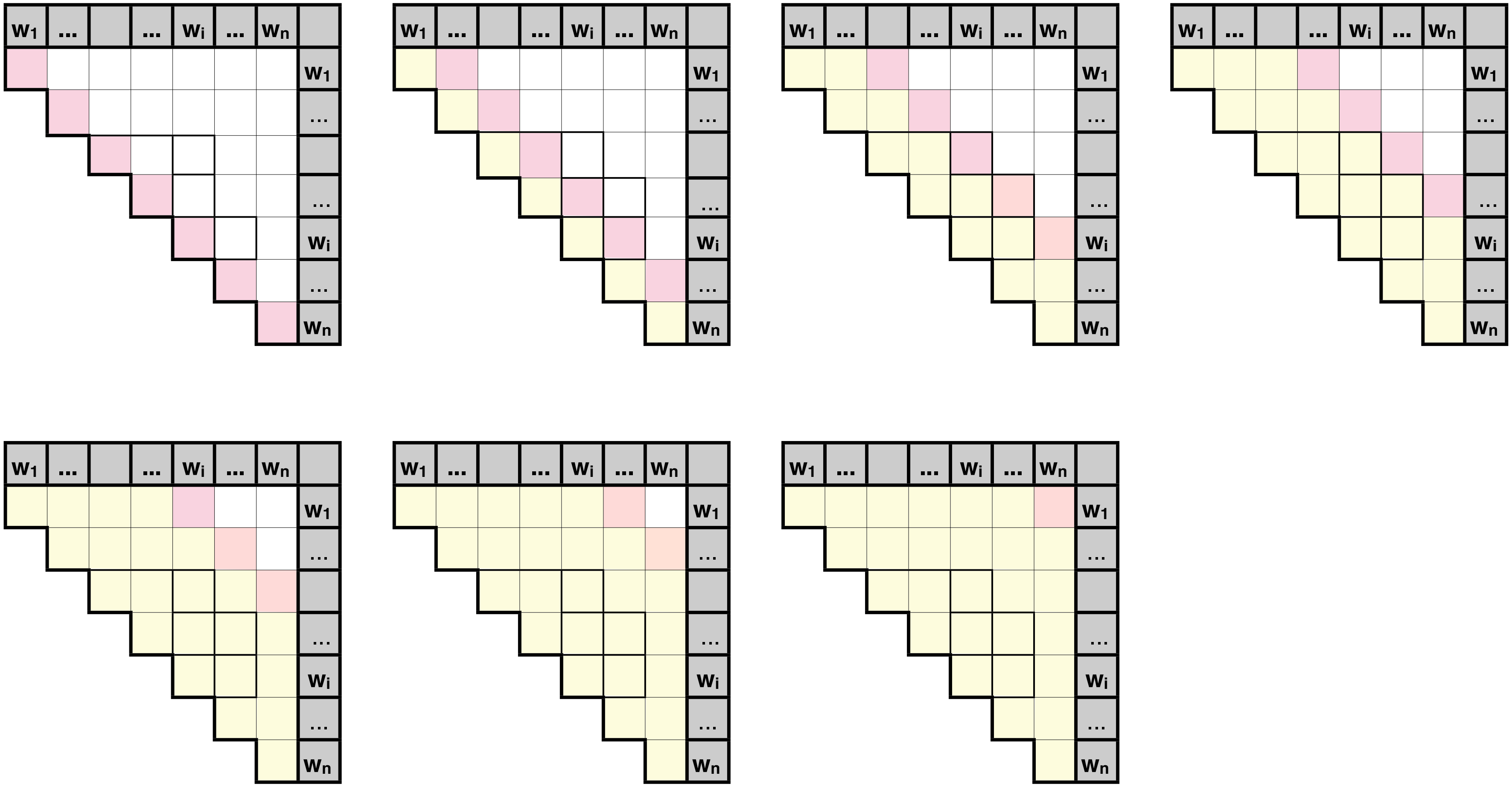
S → NP VP
VP → V NP
VP → VP PP
V → drinks
NP → NP PP
NP → we
NP → drinks
NP → milk
PP → P NP
P → with



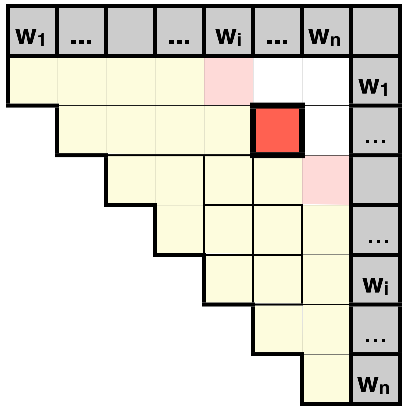
Each cell may have **one entry for each nonterminal**

We buy drinks with milk

CKY: filling the chart



CKY: filling one cell

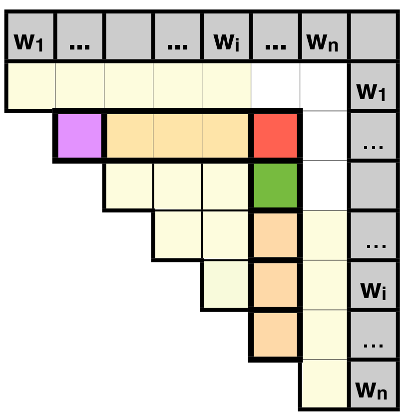


chart[2][6]:

w_1 **w_2 w_3 w_4 w_5 w_6** w_7

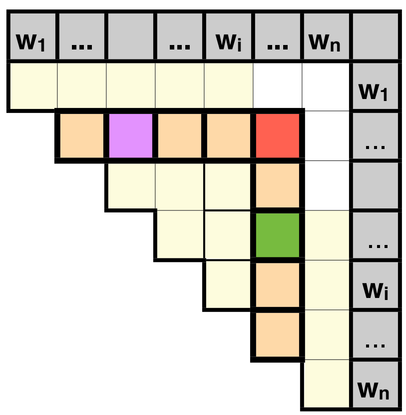
chart[2][6]:

w_1 $w_2 w_3 w_4 w_5 w_6$ w_7



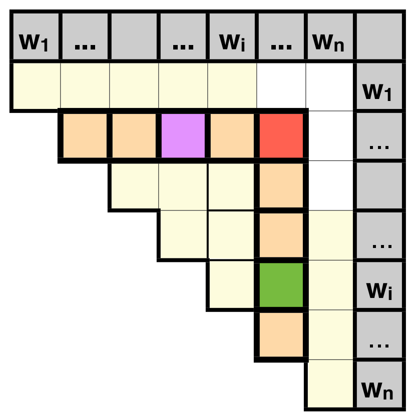
chart[2][6]:

w_1 $w_2 w_3 w_4 w_5 w_6$ w_7



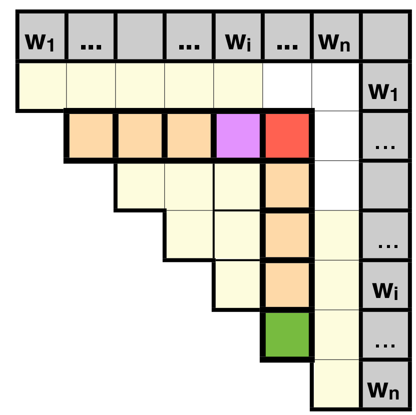
chart[2][6]:

w_1 $w_2 w_3 w_4 w_5 w_6$ w_7



chart[2][6]:

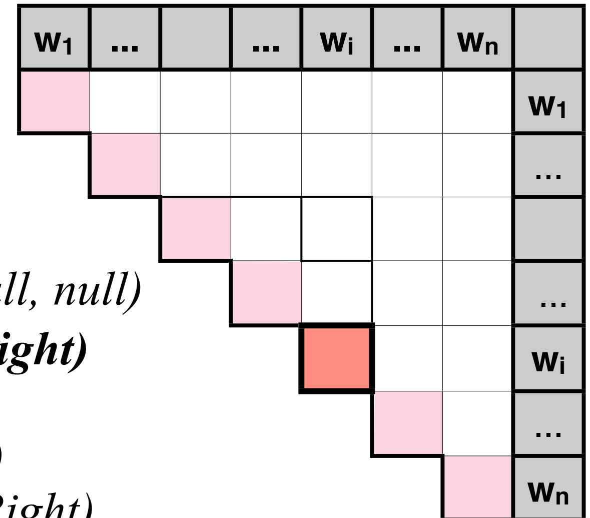
w_1 $w_2 w_3 w_4 w_5 w_6$ w_7



Cocke Kasami Younger (1)

```
ckyParse(n):
    initChart(n)
    fillChart(n)
```

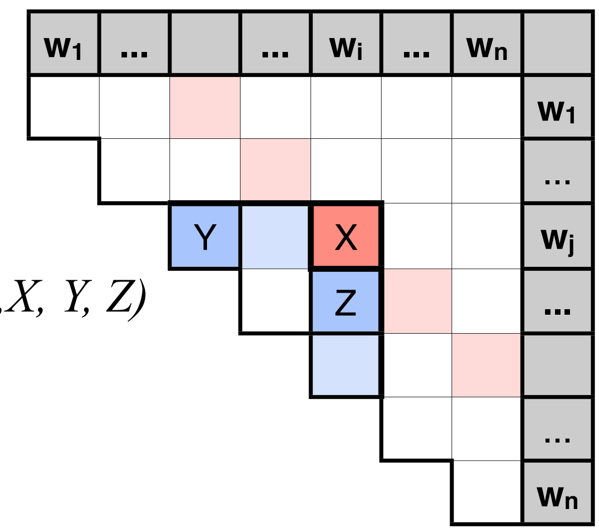
```
initChart(n):
  for i = 1...n:
      initCell(i,i)
initCell(i,i):
  for c in lex(word[i]):
      addToCell(cell[i][i], c, null, null)
addToCell(Parent,cell,Left, Right)
    if (cell.hasEntry(Parent)):
        P = cell.getEntry(Parent)
        P.addBackpointers(Left, Right)
  else cell.addEntry(Parent, Left, Right)
```




```
fillChart(n):
  for span = 1...n-1:
      for i = 1...n-span:
          fillCell(i,i+span)

fillCell(i,j):
  for k = i..j-1:
    combineCells(i, k, j)
```

```
combineCells(i,k,j):
  for Y in cell[i][k]:
    for Z in cell[k +1][j]:
      for X in Nonterminals:
        if X →Y Z in Rules:
          addToCell(cell[i][j],X, Y, Z)
```



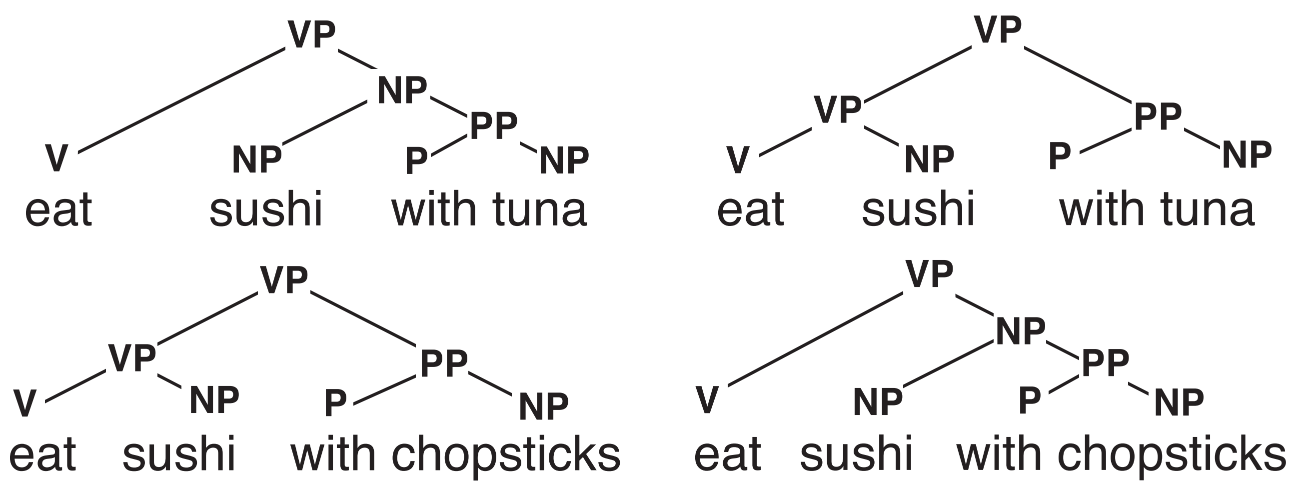
Exercise: CKY parser

$$
\begin{aligned}
\mathbf{S} &\rightarrow \mathbf{NP\ VP} \\
\mathbf{NP} &\rightarrow \mathbf{NP\ PP} \\
\mathbf{NP} &\rightarrow \mathbf{Noun} \\
\mathbf{VP} &\rightarrow \mathbf{VP\ PP} \\
\mathbf{VP} &\rightarrow \mathbf{Verb\ NP}
\end{aligned}
$$

I eat sushi with chopsticks

Grammars are ambiguous

A grammar might generate multiple trees for a sentence:



What's the most likely parse τ for sentence S ?

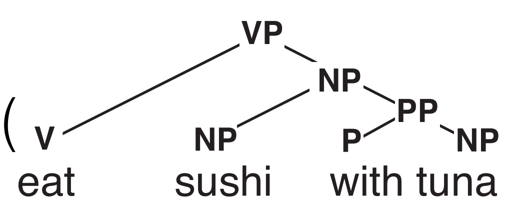
We need a model of $P(\tau \mid S)$

Computing P(τ | S)

Using Bayes' Rule:

$$
\begin{aligned}
\arg\max_{\tau} P(\tau|S) &= \arg\max_{\tau} \frac{P(\tau, S)}{P(S)} \\
&= arg\max_{\tau} P(\tau, S) \\
&= arg\max_{\tau} P(\tau) \quad \text{if} \quad \mathrm{S} = \mathrm{yield}(\tau)
\end{aligned}
$$

The **yield of a tree** is the string of terminal symbols that can be read off the leaf nodes

yield(

) = *eat sushi with tuna*

Computing P(τ)

T is the (infinite) set of all trees in the language:

$$L = \{s \in \Sigma^* \mid \exists \tau \in T : \text{yield}(\tau) = s\}$$

We need to define P(τ) such that:

$$\forall \tau \in T : \quad 0 \leq P(\tau) \leq 1$$

$$\sum_{\tau \in T} P(\tau) = 1$$

The set T is generated by a context-free grammar

S	$\longrightarrow$	NP VP	VP	$\longrightarrow$	Verb NP	NP	$\longrightarrow$	Det Noun
S	$\longrightarrow$	S conj S	VP	$\longrightarrow$	VP PP	NP	$\longrightarrow$	NP PP
S	$\longrightarrow$	.....	VP	$\longrightarrow$	.....	NP	$\longrightarrow$	.....

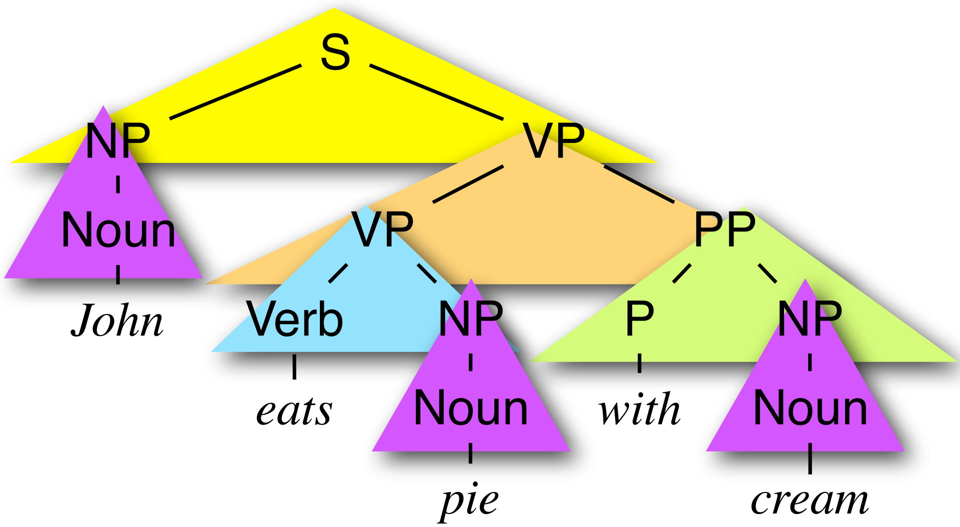
Probabilistic Context-Free Grammars

For every nonterminal X, define a probability distribution $P(X \rightarrow \alpha \mid X)$ over all rules with the same LHS symbol X:

S	$\longrightarrow$	NP VP	0.8
S	$\longrightarrow$	S conj S	0.2
NP	$\longrightarrow$	Noun	0.2
NP	$\longrightarrow$	Det Noun	0.4
NP	$\longrightarrow$	NP PP	0.2
NP	$\longrightarrow$	NP conj NP	0.2
VP	$\longrightarrow$	Verb	0.4
VP	$\longrightarrow$	Verb NP	0.3
VP	$\longrightarrow$	Verb NP NP	0.1
VP	$\longrightarrow$	VP PP	0.2
PP	$\longrightarrow$	P NP	1.0

Computing P(τ) with a PCFG

The probability of a tree τ is the product of the probabilities of all its rules:



S	$\longrightarrow$	NP VP	0.8
S	$\longrightarrow$	S conj S	0.2
NP	$\longrightarrow$	Noun	0.2
NP	$\longrightarrow$	Det Noun	0.4
NP	$\longrightarrow$	NP PP	0.2
NP	$\longrightarrow$	NP conj NP	0.2
VP	$\longrightarrow$	Verb	0.4
VP	$\longrightarrow$	Verb NP	0.3
VP	$\longrightarrow$	Verb NP NP	0.1
VP	$\longrightarrow$	VP PP	0.2
PP	$\longrightarrow$	P NP	1.0

$$P(\tau) = 0.8 \times 0.3 \times 0.2 \times 1.0 \times 0.2^3$$

$$= \mathbf{0.00384}$$

The three basic problems for PCFGs

We observe an **output sequence** $w = w_1 \ldots w_N$.:

$\boldsymbol{w}$="she promised to back the bill"

Problem I (Likelihood): find $P(w \mid \lambda)$

Given a PCFG with parameters λ, compute **the likelihood of the observed output**, $P(\boldsymbol{w} \mid \lambda)$

Problem II (Decoding): find $\tau^* = \mathrm{argmax}\ \mathrm{P}(\tau, \mathrm{w} \mid \lambda)$

Given a PCFG with parameters λ, what is **the most likely tree** τ to generate $\boldsymbol{w}$?

Problem III (Estimation): find $argmax_{\lambda}\ P(w \mid \lambda)$

Find the parameters λ which maximize $P(\boldsymbol{w} \mid \lambda)$

PCFG parsing (decoding): Probabilistic CKY

Probabilistic CKY

Like standard CKY, but with probabilities.

Terminals have probability $p = 1$

Non-terminals: associate $P(X \rightarrow YZ \mid X)$ with every pair of backpointers from `X` in `cell[i][j]` to `Y` in `cell[i][k]` and `Z` in `cell[k+1][j]`

Finding the most likely parse

Local greedy (Viterbi) search is guaranteed to be optimal:
For every non-terminal `X` in `cell[i][j]`, keep only the highest-scoring pair of backpointers to any pair of children (`Y` in `cell[i][k]` and `Z` in `cell[k+1][j]`):

$$P_{VIT}(X) = \boldsymbol{argmax}_{Y,Z}\, P_{VIT}(Y) \times P_{VIT}(Z) \times P(X \rightarrow YZ \mid X)$$

Probabilistic CKY

Input: POS-tagged sentence

`John_N eats_V pie_N with_P cream_N`

John	eats	pie	with	cream	
N NP 0.2	S 0.8*0.2*0.4	S 0.8*0.2*0.08		S 0.2*0.0024*0.8	**John**
	V VP 0.4	VP 0.3*0.2		VP max(0.008*0.2, 0.06*0.2*0.2)	**eats**
		N NP 0.2		NP 0.2*0.2*0.2	**pie**
			P	PP 1*0.2	**with**
				N NP 0.2	**cream**

S	$\longrightarrow$	NP VP	0.8
S	$\longrightarrow$	S conj S	0.2
NP	$\longrightarrow$	Noun	0.2
NP	$\longrightarrow$	Det Noun	0.4
NP	$\longrightarrow$	NP PP	0.2
NP	$\longrightarrow$	NP conj NP	0.2
VP	$\longrightarrow$	Verb	0.4
VP	$\longrightarrow$	Verb NP	0.3
VP	$\longrightarrow$	Verb NP NP	0.1
VP	$\longrightarrow$	VP PP	0.2
PP	$\longrightarrow$	P NP	1.0