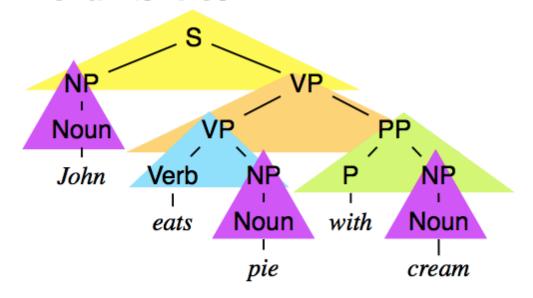
Probabilistic CFG

- normalization
 - sum_{\beta} $p(A \rightarrow \beta) = I$
- where are these probs from?
 - "gold-stand" trees -- Treebank
 - $p(A \rightarrow \beta) = \#(A \rightarrow \beta) / \#(A)$
- what's the most likely tree? easy
- what's the most likely string? hard
- given string w, what's the most likely tree for w?
 - this is called "parsing" (like decoding)

```
0.8
       \rightarrow NP VP
       \rightarrow S conj S
                                0.2
       \rightarrow Noun
                                0.2
NP
       \rightarrow Det Noun
                                0.4
NP
       \rightarrow NP PP
                                0.2
NP
                                0.2
       \rightarrow NP conj NP
NP
VΡ
       \rightarrow Verb
                                0.4
                                0.3
VP
       \rightarrow Verb NP
                                0.1
VP
       \rightarrow Verb NP NP
                                0.2
VP
       \rightarrow VP PP
       \rightarrow P NP
                                1.0
PP
```

Probability of a tree

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



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

= 0.00384

S	ightarrow NP VP	0.8
S	ightarrow S conj S	0.2
NP	ightarrow Noun	0.2
NP	ightarrow Det Noun	0.4
NP	ightarrow NP PP	0.2
NP	ightarrow NP conj NP	0.2
۷P	ightarrow Verb	0.4
VP	ightarrow Verb NP	0.3
۷P	ightarrow Verb NP NP	0.1
VP	ightarrow VP PP	0.2
PP	\rightarrow P NP	1.0

Most Likely Tree - Knuth 77

- highest-probability tree out of a PCFG
- idea I: dynamic programming (DP)
 - optimize for each nonterminal X
 - problem: cyclic updates
 - if S -> NP VP and VP -> VB S

```
S
       \rightarrow NP VP
                                 0.8
                                 0.2
       \rightarrow S conj S
                                 0.2
NP
       \rightarrow Noun
                                 0.4
NP
       \rightarrow Det Noun
                                 0.2
NP
       \rightarrow NP PP
                                 0.2
NP
       \rightarrow NP conj NP
VΡ
       \rightarrow Verb
                                 0.4
                                 0.3
       \rightarrow Verb NP
VΡ
       \rightarrow Verb NP NP
                                 0.1
VΡ
                                 0.2
       \rightarrow VP PP
VP
       \rightarrow P NP
PP
                                 1.0
```

- can't use Viterbi (which relies on topological ordering)
- idea 2: best-first DP: Dijkstra! (b/c prob <= 1, cf. non-negative)
 - every time choose the best nonterminal in the queue to expand
 - but not always possible to combine w/ others to update LHS
 - update LHS only when all RHS nonterminals are ready (popped)

Knuth 77 Example

- initial queue = (NP: 0.4, VP: 0.4)
- say pop VP: 0.4
 - which rules can be used for updates?
 - S -> NPVP 0.8; VP -> VP PP 0.2
 - can we use these rules now?
 - No, b/c NP and PP are not ready
- next, pop NP: 0.4
 - which rules can be used for updates?
 - S -> NPVP 0.8; PP ->P NP 1.0
 - update S to be 0.8x0.4x0.4=0.128 and PP to be 1.0x0.4=0.4
- next, pop PP: 0.4; but can't update anything (NP/VP already popped)
- next, pop S: 0.128, and finishes here since S is the start symbol

```
\rightarrow NP VP
                               0.8
                               0.2
       \rightarrow S conj S
                               0.2
NP
       \rightarrow Noun
                               0.4
NP
       \rightarrow Det Noun
                               0.2
NP
       \rightarrow NP PP
                               0.2
NP
      \rightarrow NP conj NP
VP
       \rightarrow Verb
                               0.4
                               0.3
       \rightarrow Verb NP
VΡ
VP
       \rightarrow Verb NP NP
                               0.1
                               0.2
VP \rightarrow VP PP
PP
      \rightarrow P NP
                                1.0
```

Most likely tree given string

- parsing is to search for the best tree t* that:
 - $t^* = argmax_t p(t | w) = argmax_t p(t) p(w | t)$
 - $= \operatorname{argmax}_{\{t: y \in Id(t) = w\}} p(t)$
 - analogous to HMM decoding
- is it related to "intersection" or "composition" in FSTs?

CKY Algorithm

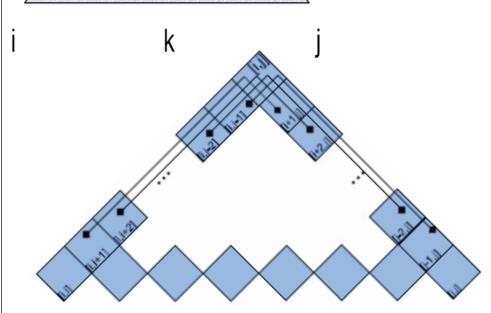
- For each diff (<= n)</p>
 - For each i (<= n)</p>
 - For each rule X → Y Z
 - For each split point k

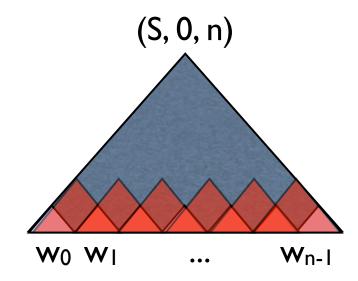
```
score[X][i][j] = max score[X][i][j],
```

score(X->YZ) *

score[Y][i][k] '

score[Z][k][j]





CKY Algorithm

- For each diff (<= n)</p>
 - For each i (<= n)</p>
 - For each rule X → Y Z
- For each split point k score[X][i][j] = max score[X][i][j], flies like flower

```
score[Y][i][k] *
                score[Z][k][j]
   S \rightarrow NPVP
NP → DT NN
                       VB \rightarrow flies
                       NNS → flies
NP \rightarrow NNS
                       VB → like
NP \rightarrow NP PP
                       P \rightarrow like
VP \rightarrow VB NP
                       DT \rightarrow a
VP \rightarrow VP PP
                       NN → flower
VP \rightarrow VB
```

 $PP \rightarrow P NP$

score(X->YZ) *

CKY Algorithm

- For each diff (<= n)</p>
 - For each i (<= n)</p>
 - For each rule X → Y Z
- For each split point k score[X][i][j] = max score[X][i][j], score(X->YZ) * score[Y][i][k] * 76.66 score[Z][k][j] $S \rightarrow NPVP$ $NP \rightarrow DT NN$ $VB \rightarrow flies$ 45,181 18.6. NNS → flies $NP \rightarrow NNS$ VB → like $NP \rightarrow NP PP$ $P \rightarrow like$ $VP \rightarrow VB NP$ $DT \rightarrow a$ $VP \rightarrow VP PP$

flower

 $VP \rightarrow VB$

 $PP \rightarrow P NP$

note: unary rules

like

flies

NN → flower

CKY Example

Input: POS-tagged sentence

John_N eats_V pie_N with_P cream_N

Jo	ohn	ea	ats	p	oie	with	cream		
N	NP 0.2	S 0.8*0.2*0.4					S 0.2*0.0024*0.8		John
		٧	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
			'			Р		PP 1*0.2	with
					'		N	NP 0.2	cream

S	\rightarrow NP VP	0.8
S	ightarrow S conj S	0.2
NP	ightarrow Noun	0.2
NP	ightarrow Det Noun	0.4
NP	\rightarrow NP PP	0.2
NP	ightarrow NP conj NP	0.2
۷P	ightarrow $ extstyle extstyle$	0.4
۷P	ightarrow Verb NP	0.3
۷P	ightarrow Verb NP NP	0.1
VP	\rightarrow VP PP	0.2
PP	\rightarrow P NP	1.0

Chomsky Normal Form

- wait! how can you assume a CFG is binary-branching?
- well, we can always convert a CFG into Chomsky-Normal Form (CNF)
 - \bullet A \rightarrow B C
 - \bullet A \rightarrow a
- how to deal with epsilon-removal?
- how to do it with PCFG?